(2)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY VLSI PUBLICATIONS





VLSI Memo No. 89-534
May 1989

# A Digital Model for Level-Clocked Circuitry

Alexander Toichi Ishii

Abstract

This thesis presents the formal background for a mathematical model for level-clocked circuitry, in which latches are controlled by the levels (high or low) of clock signals rather than transitions (edges) of the clocks. Such level-clocked circuits are frequently used in MOS VLSI design. Our model maps continuous data-domains, such as voltage, into discrete, or *digital*, data domains, while retaining a continuous notion of time. A level-clocked circuit is represented as a graph $G = (V,E)$, where $V$ consists of digital components--latches and functional elements--and $E$ represents inter-component connections.

The majority of this thesis concentrates on developing lemmas and theorems that can serve as a set of "axioms" when analyzing algorithms based on the model. Key axioms include the fact that circuits in our model generate only well defined digital signals, and the fact that components in our model support and accurately handle the "undefined" values that electrical signals must take on when they make a transition between valid logic levels. In order to facilitate proofs for circuit properties, the class of *computational predicates* is defined. A circuit property can be proved by simply casting the property as a computational predicate.





Microsystems Research Center
Room 39-321
Massachusetts Institute of Technology
Cambridge Massachusetts 02139
Telephone (617) 253-8138

Acknowledgements

Submitted to the Department of Electrical Engineering and Computer Science, MIT, August 1988, in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering and Computer Science. This work was sponsored in part by the Defense Advanced Research Projects Agency under contract number N00014-87-K-0825.

Author Information

Ishii: Department of Electrical Engineering and Computer Science, Room NE43-338, MIT, Cambridge, MA 02139. (617) 253-7583.

# A Digital Model for Level-Clocked Circuitry

by
Alexander Toichi Ishii

Submitted to the Department of
Electrical Engineering and Computer Science
on August 8, 1988
in Partial Fulfillment of the
Requirements of the Degree of

Master of Science
in
Electrical Engineering and Computer Science



**Abstract**

This thesis presents the formal background for a mathematical model for level-clocked circuitry, in which latches are controlled by the levels (high or low) of clock signals rather than transitions (edges) of the clocks. Such level-clocked circuits are frequently used in MOS VLSI design. Our model maps continuous data-domains, such as voltage, into discrete, or *digital*, data domains, while retaining a continuous notion of time. A level-clocked circuit is represented as a graph $G = (V, E)$, where $V$ consists of digital components—latches and functional elements—and $E$ represents inter-component connections.

The majority of this thesis concentrates on developing lemmas and theorems that can serve as a set of "axioms" when analyzing algorithms based on the model. Key axioms include the fact that circuits in our model generate only well defined digital signals, and the fact that components in our model support and accurately handle the "undefined" values that electrical signals must take on when they make a transition between valid logic levels. In order to facilitate proofs for circuit properties, the class of *computational predicates* is defined. A circuit property can be proved by simply casting the property as a computational predicate.

Thesis Supervisor: Professor Charles E. Leiserson
Title: Associate Professor

[0]This research represents joint work with Charles E. Leiserson, and is supported in part by the Defense Advanced Research Projects Agency under Contract N00014-87-K-825. Charles Leiserson is supported in part by an NSF Presidential Young Investigator Award with matching funds provided by AT&T Bell Laboratories and Xerox Corporation.

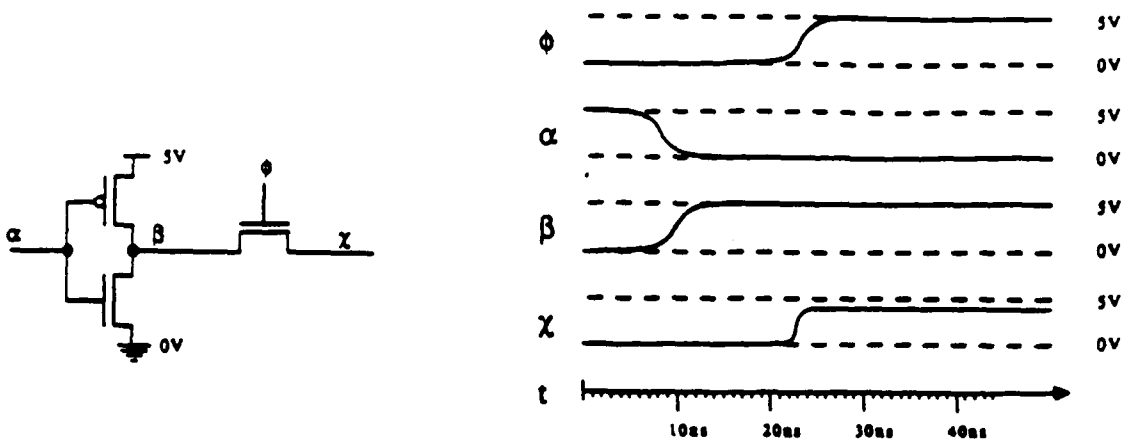


**Figure 1**: Shown is a transistor-level circuit diagram for a simple cMOS level-clocked circuit. Labels on electrical nodes indicate the time dependent voltages computed for them by the device-level simulator SPICE.

# 1 Introduction

The MOS/VLSI technology has popularized a methodology of clocking based on level-clocked latches instead of the more traditional edge-triggered latches used, for example, in TTL design. The popularity of level-clocking arises from the simplicity with which a level-clocked latch can be implemented in MOS technologies: a single transistor can suffice. Unfortunately, the high device densities being achieved with modern VLSI fabrication processes preclude the possibility of performing detailed circuit simulations of complete level-clocked circuit systems. Consequently, the design and analysis of level-clocked circuit systems require models that can sacrifice detail, while maintaining accuracy.

The lowest level at which level-clocked circuits are modeled is commonly referred to as the *device level*. At this level, small circuits consisting of at most a few dozen electrical devices are simulated in great detail, according to empirically verified models for individual device behavior. In general, signals being passed between devices are time-dependent voltages that can take on values in some continuous range. Thus, within the limitations of floating-point number representation, device-level models treat signals as mappings from continuous time to some continuous data domain, generally voltage. Figure 1 shows the output of the device-level simulator SPICE [16], for a simple cMOS level-clocked circuit.

The next two levels at which level-clocked circuits are commonly modeled are referred to as the *switch level* and *block level*. At these levels, large circuits of perhaps thousands of individual devices are simulated in an attempt to uncover difficulties with circuit functionality and data-movement coordination. Like device-level models, signals are generally still mappings from continuous time to some data domain. At these levels of representation, however, the data domains of signals are abstracted to be discrete, or *digital*, rather than continuous. Figure 2 depicts a block level representation of the level-clocked circuit from figure 1.

The reason for the abstraction to digital data domains is two fold. First, the abstrac-

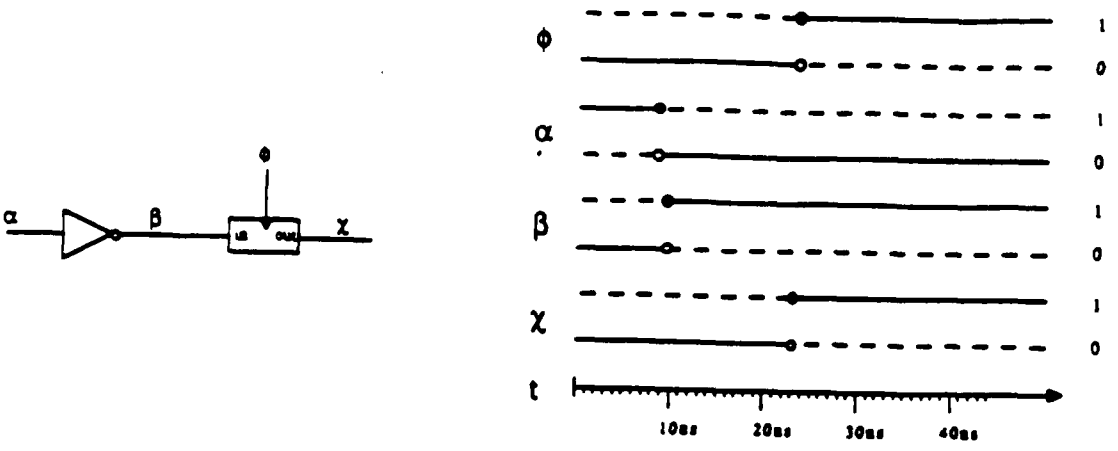


**Figure 2**: Shown is a block level representation of the level-clocked circuit from figure 1, along with its corresponding digital signals. Note that two of the transistors from figure 1 have been grouped into a single "block" of combinational logic, represented as a logical inverter, and the remaining transistor has been replaced with an abstract level-clocked latch.

tion allows the use of less detailed, and therefore less computationally intensive, device models, and consequently facilitates the simulation of circuits that contain large numbers of electrical devices. Second, the abstraction is a natural one, since the vast majority of level-clocked circuits are used in digital computers, where various voltage ranges are by convention interpreted as a logical "1" or "0."

Unfortunately, the abstraction to digital signals can be problematic, since we wish switch and block level models to accurately reflect the operation of a large level-clocked circuit. In this context, it is not enough for a model to map an electrical signal into a digital data-domain in a reasonable fashion. A model must also be able to determine whether the presumably unmodeled device-level behavior of the level-clocked circuit is such that a particular mapping would be appropriate.

Traditionally, the models used at the block and switch levels have focused more on handling the parameters of a general engineering situation, and less on formal properties. Consequently, the problem of confirming the accuracy of the mapping of electrical signals into the digital data domain has generally either been deferred to the electrical engineer, or ignored. In addition, the relative de-emphasis on formal properties has resulted in models and algorithms that lack the kinds of theoretically rigorous notions, algebras and bounds that have been developed for circuits utilizing edge-triggered latches[9, 11].

This thesis presents the formal background for a model for level-clocked circuitry that has been formulated explicitly to support mathematically precise manipulation, while maintaining the ability to accurately map electrical signals. Features of the model include the ability to support and handle the "undefined" values that electrical signals take on when they change between valid logic levels, and the ability to support formal proof techniques, such as induction. We show that circuits in the model always generate signals that have well defined values, and present several lemmas that formally characterize the behavior of

individual circuit components. In addition, we define a large class of circuit properties, the *computational predicates*, and in conjunction develop the mechanism of *computational ordering*, which conveniently integrates the limit arguments used to analyze continuous-time systems into a more easily manipulated inductive framework.

The remainder of this thesis is organized as follows. Section 2 defines the basic concepts of digital signals and models. Section 3 presents our models for individual circuit components, and proves the ability of individual components to accurately operate on digital signals. Section 4 defines how circuit components are hooked up into complete circuits, and presents our representation of a computation on a circuit. Section 5 develops the mechanism of computational ordering, and defines the class of computational predicates. Section 6 then uses computational predicates to prove that circuits in the model possess important basic properties, such as deterministic operation. Finally, section 7 finishes with some concluding remarks that do not belong in the introduction.

## 2 Digital Signals and Models

In this section we define *digital signals* and their correspondence to the *electrical signals* that they are used to represent. Mathematical definitions are given along with intuitive descriptions when appropriate. In addition, we introduce the concept of a digital model.

A tacit assumption throughout this thesis is that we wish to accurately represent the *electrical* signals found in real MOS circuits, without resorting to the powerful, but computationally intensive, device-level simulations of systems such as SPICE. Electrical signals generally are time-dependent electrical voltages or currents that vary over some continuous range of values. They are determined either empirically, or with complex nonlinear device models. The assumption throughout this thesis is that electrical signals are represented by time dependent signals that only vary over some *digital* data-domain, which contains only a countable number of values. A *digital model* is a model that only handles signals with digital data domains.

When signals with digital data domains are used to represent electrical signals, there must exist some correspondence of values in the data domain of the electrical signals, to values in the digital data domain. Electrical signals in MOS circuits, for example, might use voltages greater than or equal to 3.5 volts, and less than or equal to 1.5 volts to represent binary 1 and 0 respectively. The correspondence of values in the data domain of the electrical signals to values in the digital data domain is the *elementary mapping* for the data domain of the electrical signals.

Since digital models are presumably less powerful than device-level models, we expect that there are times when a digital model is not powerful enough to accurately perform the elementary mapping. Such times can be grouped into two broad categories. The first category consists of times when the elementary mapping does not specify a value in the digital data-domain, that for the value of an electrical signal. At such times the value of a digital signal is undeterminable, because the value that corresponds to the electrical signal is *undefined* in the digital data domain. The second category consists of times when the elementary mapping does specify a value in the digital data domain, for the value of an electrical signal, but the digital model does not incorporate sufficient detail to perform the mapping. At such times, the value of a digital signal is undeterminable, because the correct values are effectively *underdefined* by the digital model.

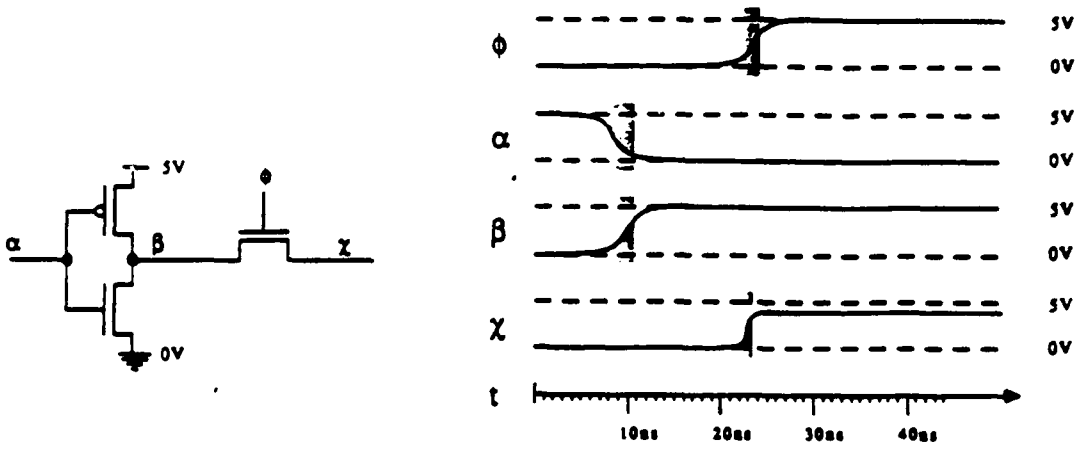


**Figure 3**: Shown is the level-clocked circuit from figure 1, and its time dependent voltage waveforms. Times that correspond to undefined values are shaded.

Undefined values are generally due to the fact that we are mapping the the value of some continuous physical quantity into the discrete data domain of some digital signal. For example, consider the previously stated voltage range elementary mapping for MOS circuits. Since voltage is a continuous physical quantity, an electrical signal cannot change from 3.5 to 1.5 volts, without taking on every intermediate voltage. Times when the electrical signal is at some intermediate voltage cannot be mapped to either 1 or 0, and consequently correspond to a value that is undefined in the digital data-domain. Figure 3 shows the times that correspond to undefined values for the circuit from figure 1.

Underdefined values can be due to uncertainty inherent to the MOS circuits being modeled. For example, it is often impossible to predict the exact time at which an electrical signal would change value, since variations in circuit fabrication processing make it difficult to predict the amount of time needed for a change in voltage to propagate from one piece of circuitry to another. Consequently, a transition in the value of the electrical signal, implies an underdefined interval during which an accurate representation of electrical signal in a digital data domain is not possible. In general, this interval is made large enough to encompass all times at which the transition might actually occur.

In general, however, underdefined values are due to the desire to avoid computing the solutions to computation intensive problems. For example, in section 3, it will be assumed that a change in the value of an input to a circuit component, forces the output of the component to be considered underdefined. This constraint is excessive, since there are circuits, such as a typical nMOS NOR gate, where inputs can sometimes change value without affecting the value of the output. Unfortunately, the general problem of determining whether a circuit falls into this category is NP-complete [3], and consequently is likely to be computationally intractable. Thus, while changing inputs do not actually imply that the output of a circuit cannot be mapped into a digital data domain, the output is effectively underdefined, since modern computing machinery cannot compute the mapping efficiently.

In order to model underdefined and undefined values, we introduce into every digital data-domain the special symbol $\perp$, which is specifically used to represent undefined and underdefined values. The explicit representation of underdefined and undefined values is important, since, as we shall see in section 3, underdefined and undefined values have a significant impact on how circuits behave.

A digital signal is essentially a mapping from time to some discrete, or *digital*, data domain $\mathcal{D}$, typically $\{0,1\}$, where $\mathcal{D}$ does not contain the reserved symbol $\perp$. Elements of $\mathcal{D}$ are *valid* digital values, while the symbol $\perp$ is the *invalid* digital value that we use to represent underdefined and undefined values. Each element of $\mathcal{D}$ is assumed to correspond to some subset of the continuous data domain for the modeled electrical signal, just as "1" and "0" correspond to voltage ranges in the description of figure 3. We also assume, however, that as the value of an electrical signal changes from a subset that corresponds to one *valid* value, to a subset that corresponds to another, there exists an interval of time during which its value does not correspond to any element of $\mathcal{D}$. During such intervals, the value of a digital signal is the invalid value $\perp$. The following mathematical definition formalizes this characterization.

**Definition.** A *digital signal* over a discrete data-domain $\mathcal{D}$ is a mapping $s : \mathbb{R} \cup \{-\infty\} \rightarrow \mathcal{D} \cup \{\perp\}$, that satisfies the following two properties.

- For each $x \in \mathcal{D}$, the values $t$, such that $s(t) = x$, form a set $\textsc{Stable}(s, x)$ of nonoverlapping closed intervals.
- For each $x \in \mathcal{D}$, the set $\textsc{Stable}(s, x)$ is *locally finite*, that is, for all $t_1, t_2 \in \mathbb{R}$, the number of intervals in $\textsc{Stable}(s, x) \cap [t_1, t_2]$ is finite.

Observe, that when an digital signal changes from one valid value to another, the first property implies that there exists a well defined last moment in time when it takes on the first value, a well defined first moment in time when it takes on the second value, and an open interval of time between them where it takes on the invalid value $\perp$. In addition, the second property excludes any signal whose value changes infinitely often within a finite period of time, thus guaranteeing that there exists an order-preserving mapping from stable intervals to the integers. The order preserving mapping to the integers is of great significance in section 5, where we perform inductions on digital signals. Finally, while we expect the data domain for signals to be circuit dependent, for convenience we generally assume that all signals in a circuit share a common data domain $\mathcal{D}$.

We say a digital signal is *stable* over an interval of time if it assumes a constant valid value over the interval. By definition, a stable signal is constant. A constant signal need not be stable, however, since a signal could be constant with the invalid value $\perp$. We specifically use the term *constant* to indicate that a signal that could also be constant with value $\perp$. Observe, that by definition, a digital signal is stable over any interval that is an element of $\textsc{Stable}(s, x)$, for some $x \in \mathcal{D}$.

# 3 Ideal Circuit Components

This section presents our model for individual circuit components. We begin with an overview of what circuit components are, and how our model represents circuit component behavior. Then for each type of circuit component we present an intuitive description of

the behavior of the component, describe how underdefined and undefined values affect this behavior and present our mathematical model for the component. In addition, several basic lemmas are presented. These lemmas serve as a basic set of "axioms" for subsequent theorems, and their proofs provide a valuable opportunity to become familiar with different aspects of our models for circuit components.

In the broadest sense, a circuit component is anything that can be used as part of a complete circuit. At the device level, common components are wires, transistors, resistors and capacitors. At the block level, it is common to encounter more abstract components such as logic gates and ALUs. In general, the behavior of a circuit component at the block level is defined by the relationship between the inputs and output of the component, and not by the physical objects with which the component is realized.

Our model groups circuit components into two basic categories. The first category includes the *latches* that control the movement of data within a circuit. The behavior of latches is carefully represented in our model. The second category includes all components that are not latches. This category includes everything from device-level components such as transistors, to block-level components such as logic gates. Components in this category are called *functional elements*. Since the behavior of functional elements can range between the extremely simple and the very complex, our model represents their behavior in a generic fashion.

Latches are placed in their own category for two reasons. First, since latches control data movement within a circuit, they are the components of primary concern when performing circuit verification operations, such as timing-analysis [2, 4, 5, 7, 12, 15]. Second, while most types of functional elements, such as transistors or logic gates, are grouped into larger components at higher levels of abstraction, latches often remain atomic and exhibit essentially the same behavior. Consequently, latches enjoy the unique position of being considered important fundamental components at almost all levels of circuit modeling.

Our model represents circuit components with *digital circuit components*. An digital circuit component is an abstract object that has some number $k$ of digital input-signals $x_1, x_2, \ldots, x_k$, a single digital output-signal $y$, and a constraint

$$\textsc{Legal}(y, x_1, x_2, \ldots, x_k, t)$$

that if satisfied at time $t$ indicates, that the input signals and the output signal are consistent with the behavior of the component. Observe that since arbitrary digital signals can be used as the inputs and outputs of digital components, digital components do not generate outputs in the traditional sense. In general we assume that input and output signals are given and specified over all time. Under this assumption, the issue is not "what" the value of a particular digital signal is, but rather whether a set of digital signals satisfies the constraint LEGAL when its elements are used as the input and output signals of some digital component.

We separately examine the digital components used to model functional elements and latches in the following two subsections. Definitions of the LEGAL constraints are given, along with intuitive descriptions where appropriate.

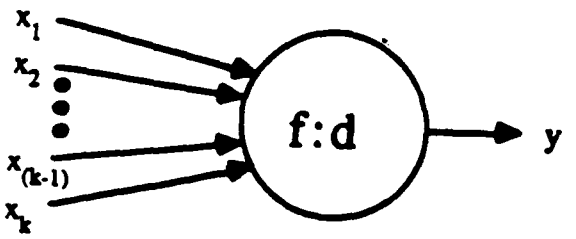


**Figure 4**: A digital functional element has some finite number $k$ of digital input signals $x_1$ thru $x_k$, a single digital output signal $y$, an associated $k$-input function $f$, and a propagation delay $d$. When an input changes, the output must immediately be $\perp$ and cannot be a valid value until a time equal to the propagation delay $d$ after the change in the input.

### 3.1 Functional Elements

At an intuitive level, a functional element is a component whose output signal is some function $f$ of its input signals. In addition, a functional element has associated with it a "settling" time, or *propagation delay* $d$, that indicates the amount of time required for the output to assume its correct value after an input changes. This propagation delay in general varies with the particular input involved, and with the specific value that the input changed to or from.

Underdefined and undefined inputs to functional elements represent special difficulties, due to the wide variety of circuit components that get grouped as functional elements. Intuitively, an input to a functional element being underdefined or undefined, should imply that the output of the functional element is also. Similarly, the output of a functional element should be underdefined or undefined for at least one propagation delay after a change in the value of some input signal. As mentioned in section 2, however, there exist circuits whose outputs can be well defined, even if one or more inputs are not. Unfortunately, verifying that an arbitrary functional element falls into this category is likely to be too computationally expensive to be practical.

We model functional elements with a digital component. A *digital functional element* has some finite number $k$ of digital input signals $x_1$ thru $x_k$, a single digital output signal $y$, an associated $k$-input function $f$, and a propagation delay $d$, as shown in figure 4. The constraint LEGAL$(y, x_1, x_2, \ldots, x_k, t)$ for a functional element is satisfied at time $t$, if and only if the signals $y, x_1, x_2, \ldots, x_k$ satisfy the following equation:

$$y(t) = \begin{cases} f(x_1(t), x_2(t), \ldots, x_k(t)) & \text{if } x_i \text{ is stable for all } i = 1, 2, \ldots, k \\ & \text{over the interval } [t-d, t], \\ \perp & \text{otherwise.} \end{cases} \quad (1)$$

There are three features of equation 1 that should be noted. First, if an input signal changes value and the output signal is to satisfy LEGAL, the output signal must immediately be $\perp$ and cannot be a valid value until a time equal to the propagation delay $d$ after the change in the input. Second, if any input signal is $\perp$ at time $t$, by definition it is not

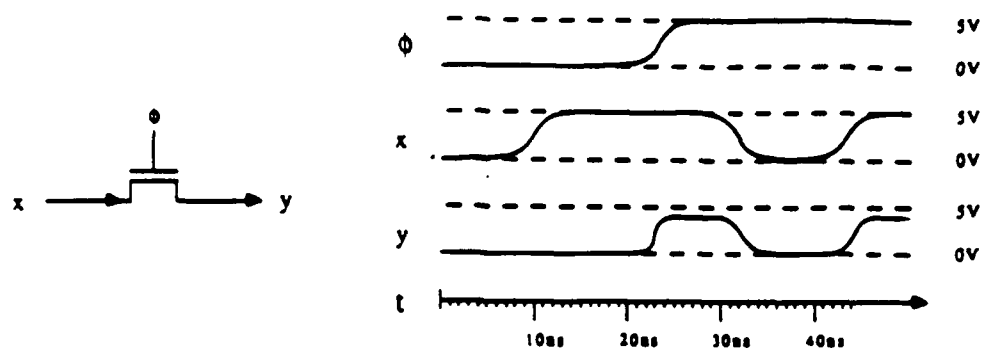


**Figure 5**: Shown is a simple circuit that implements a level-clocked latch. The latch has input signal $x$, output signal $y$ and clock signal $\phi$. When $\phi$ is high, $y$ follows $x$. When $\phi$ is low, $y$ holds the value it had when $\phi$ was most recently high.

stable at time $t$, and thus LEGAL constrains the output signal to be $\perp$. Observe that these features are equivalent to the previously described intuitive notions of underdefined and undefined values. Without these features, our model would be prone to computational intractability difficulties. The third feature to note is that digital functional elements are characterized by only a single propagation delay $d$. The fact that $d$ does not vary over the different inputs is simply for convenience. The fact that $d$ does not vary over the different values for inputs is significant, since this type of propagation delay would also introduce computational intractability problems.

Digital functional elements can be used to represent more general circuit components, much as ideal electrical components are used to model real physical devices. For example, a functional element with multiple outputs can be represented with several one-output digital functional elements. As another example, a functional element whose propagation delay varies with the input signal can be represented with a zero-delay functional element, each of whose input signals is the output signal of a functional element that computes the identity function and whose propagation delay is the input-to-output propagation delay of the original functional element.

### 3.2 Level-Clocked Latches

Level-clocked latches are latches that are controlled by the level (high or low) of a clock signal rather than a transition (edge) of the clock. Latches are three-terminal components that are used to store and propagate data. A latch takes a single input signal and a single clock signal, and produces a single output signal. A level-clocked latch has the following general behavior. While the clock for a level-clocked latch is high, the output of the latch is equal to its input. When the clock changes to low, the latch stores the value of its input and outputs this value until the the clock changes back to high. Figure 5 shows a simple implementation for a level-clocked latch, along with an illustrative set of input/output

signals. Such level-clocked latches are frequently used in MOS VLSI design.

The implications of underdefined and undefined values on the input signal of a level-clocked latch are fairly straight-forward. If the clock input of the latch is high, an undefined or underdefined value on the input signal implies an undefined or underdefined value on the output signal, since the input and output are equal. If the clock input is low, an undefined or underdefined value on the input signal implies nothing about the value on the output, since the latch is outputting the value stored from the last time the clock was high.

If our only concern were underdefined and undefined values on the input signal of a level-clocked latch, then we could describe the behavior of a digital level-clocked latch with an equation similar to the following:

$$y(t) = \begin{cases} x(t) & \text{if } \phi(t) = \textsc{High} \\ y(t_{\phi\text{-high}}) & \text{if } \phi(t) = \textsc{Low} \text{ and} \\ & t_{\phi\text{-high}} = \sup\{t' \leq t : \phi(t') = \textsc{High}\} \end{cases} \tag{2}$$

where $x$, $y$ and $\phi$ are the respectively the input signal, output signal and clock signal of the latch. Observe that no explicit reference to $\perp$ needs to occur, since the only source of $\perp$ values is the input signal $x$.

Unfortunately, equation 2 does not consider the more complex implications of underdefined and undefined values on the clock signal. If the clock signal is underdefined, then we do not know whether its value is high or low. Observe, however, that the output could be independent of the clock, and therefore well defined even if the clock signal is underdefined. Consider, for example, if the input signal held a constant value $X$ for all time. Introducing underdefined values into the clock signal would in general have no effect on the output signal. To see this, consider the possible values that the clock could have. If the clock were high, the output would have value $X$. If the clock were low, the output would be the value of the input at the last time the clock was high and, if we assume that the clock had been high in the past, the output would again have value $X$. What is needed is a model similar to equation 2. that retains a valid output value whenever the underdefined values of the clock signal cannot affect the output value.

Properly addressing the implications of undefined clock signal values would require an in-depth discussion of electrical device models and general VLSI design methodologies, that is beyond the scope of this thesis. We make the broad assumption that underdefined and undefined clock signal values can be considered to be equivalent. Any of the VLSI references [2, 13] can be consulted if verification of this assumption is desired.

Formulating a model that properly handles underdefined clock signal values is not straight forward, because of the many different cases that the model must address. For example, it is not clear apriori that an underdefined clock value immediately after a high clock value can be treated in the same way as an underdefined clock value immediately after a low clock value. Similarly, there is the question of how to treat intervals of underdefined values, where presumably the underdefined interval may represent a clock signal that in reality changed from high to low multiple times during the interval. Under what conditions can we guarantee that the output remains well defined regardless of what clock signal that the underdefined value represents?

Fortunately, we can show that the following definition for a digital latch has the qualities that we desire. A *digital latch* has a digital input signal $x$, a digital output signal $y$, and

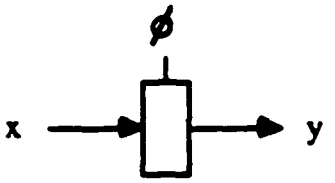


**Figure 6**: A digital latch has a single digital input signal $x(t)$, a single digital output signal $y(t)$, and an associated digital clock signal $\phi$.

a digital clock signal $\phi$, as shown in Figure 6. A digital *clock signal*, is a digital signal with the data domain {HIGH, LOW}. The constraint LEGAL$(y, x, \phi, t)$ for a ideal latch is satisfied at time $t$, if and only if the signals $y$, $x$ and $\phi$ satisfy the following equation:

$$y(t) = \begin{cases} x(t) & \text{if } \phi(t) = \text{HIGH} \\ y(t_{\phi\text{-valid}}) & \text{if } \phi(t) = \perp \text{ and } x(t') = y(t_{\phi\text{-valid}}) \\ & \text{for all } t' \in (t_{\phi\text{-valid}}, t], \\ y(t_{\phi\text{-valid}}) & \text{if } \phi(t) = \text{LOW and } x(t') = y(t_{\phi\text{-valid}}) \\ & \text{for all } t' \in (t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}), \\ \perp & \text{otherwise,} \end{cases} \tag{3}$$

where

$$t_{\phi\text{-invalid}} = \sup \{t' \leq t : \phi(t') = \perp\}$$

and

$$t_{\phi\text{-valid}} = \sup \{t' \leq t : \phi(t') \neq \perp \text{ and } \exists t'' \in (t', t] \text{ such that } \phi(t'') = \perp\}.$$

The time $t_{\phi\text{-invalid}}$ is intuitively the most recent time that the value of $\phi$ changed from $\perp$ to a valid value. Similarly, the time $t_{\phi\text{-valid}}$ is intuitively the most recent time that $\phi$ made a transition from a valid value to $\perp$. Figure 7 shows a digital clock signal $\phi$, and $t_{\phi\text{-valid}}$ and $t_{\phi\text{-invalid}}$ for $t = 25$ns and $t = 40$ns.

For the remainder of this thesis, we will be manipulating intervals that are frequently delimited by values such as $t_{\phi\text{-invalid}}$ and $t_{\phi\text{-valid}}$. Unfortunately, since the supremum of a set is not necessarily a member of the set, use of a supremum to define the start or end of an interval can lead to ambiguities with regard to whether the interval is open or closed. Rather than always explicitly enumerate the various possibilities of open, closed, half-open-below and half-open-above, we adopt the convention of using the delimiters "⟦" and "⟧" to indicate that the inclusion of the start or end point of an interval will vary from situation to situation. For example, the expression $(t_s, t_e)\rrbracket$ denotes the interval containing $(t_s, t_e)$ and possibly $t_e$. Except when noted otherwise, however, it is assumed that the inclusion or exclusion of an end point is constant for any particular interval. Consequently, given an interval $\llbracket t_s, t_e \rrbracket$, a subinterval such as $\llbracket t_s, t_e' \rrbracket$ or a derived interval such as $\llbracket t_s - x, t_e \rrbracket$ would contain $t_s$ and $t_s - x$, if and only if the original interval $\llbracket t_s, t_e \rrbracket$ contained $t_s$.

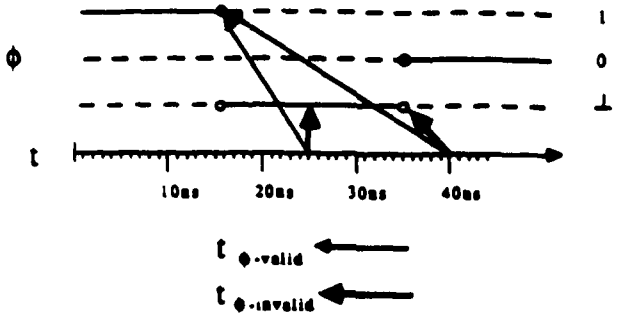


*Figure 7: Shown is a digital clock signal $\phi$, and $t_{\phi\text{-valid}}$ and $t_{\phi\text{-invalid}}$ for $t = 25$ns and $t = 40$ns.*

We can obtain an intuitive feel for equation 3, by assuming that LEGAL$(y, x, \phi, t)$ is TRUE for all times $t$, and viewing $y$ as a function of $x$ and $\phi$. When $\phi$ is HIGH, $y$ follows $x$ and the latch behaves like a functional element computing the identity function. When $\phi$ is invalid, $y$ follows $x$ as long as $x$ holds the value $y$ had at the last transition of $\phi$. If $x$ changes value while $\phi$ is invalid, the value of $y$ becomes invalid. Finally, when $\phi$ is LOW, $y$ effectively holds the value it had when $\phi$ most recently changed value to LOW. To see this, recall that $\phi$ must equal $\perp$ before becoming LOW and note that the conditions for $\phi$ equal to LOW and $\perp$ are identical until the most recent transition of $\phi$.

The following five lemmas formally verify that equation 3 is such that digital latches exhibit the types of behavior that we desire. Lemma 3.1 addresses the case where $\phi$ has value LOW, while lemmas 3.2 and 3.3 examine the more involved case where $\phi$ has value $\perp$. The case where $\phi$ has value HIGH is sufficiently simple not to warrant separate discussion. Lemmas 3.4 and 3.5 prove a pair of lemmas that are presented here for convenience, but are used in section 6.

Lemma 3.1 states that the output of a level-clocked latch must be constant when its clock input has value LOW. The lemma verifies the ability of a digital latch to hold state information, and isolate its output signal from changes in its input signal. These abilities are important, since level-clocked latches with LOW clock inputs are in general used for precisely these purposes. In addition, the lemma is needed for the proofs of lemmas 3.2 and 3.3.

**Lemma 3.1** *Let $s_{v'}$ be the input signal, $s_v$ be the output signal, and $\phi$ be the clock signal of an ideal latch, where* LEGAL$(s_v, s_{v'}, \phi, \mathcal{T})$ *for the latch is satisfied for all time $\mathcal{T}$. If $\phi$ is* LOW *over the interval $[\![t_s, t_e]\!]$, then $s_v$ must be constant over the interval $[\![t_s, t_e]\!]$.*

**Proof:** If $\phi$ is LOW over the interval $[\![t_s, t_e]\!]$, equation 3 specifies that for any time $t$ during $[\![t_s, t_e]\!]$, the value of $s_v$ is determined in one of two possible ways. Either the condition that $s_{v'}(t') = s_v(t_{\phi\text{-valid}})$, for all $t' \in (t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, is satisfied and $s_v(t) = s_v(t_{\phi\text{-valid}})$ or it is not satisfied and $s_v(t) = \perp$.

Observe, however, that by definition $t_{\phi\text{-high}}$ and $t_{\phi\text{-invalid}}$ must both be less than or equal to $t_s$, and moreover must both be constant for the entire interval $[\![t_s, t_e]\!]$. Consequently, if the condition that $s_{v'}(\mathcal{T}) = s_v(t_{\phi\text{-valid}})$ for all $\mathcal{T} \in (t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$ is satisfied at any time during $[\![t_s, t_e]\!]$ then it must be satisfied for all time during $[\![t_s, t_e]\!]$. Similarly, if the

condition is not satisfied at some time during $[\![t_s, t_e]\!]$, then it cannot be satisfied at any time during $[\![t_s, t_e]\!]$. In either case, $s_v(t)$ must be constant, either with value $s_v(t_{\phi\text{-valid}})$ or $\perp$, for all $t \in [\![t_s, t_e]\!]$. ∎

Lemmas 3.2 and 3.3 examine the behavior of a digital latch whose clock input is underdefined and $\perp$. Our eventual goal is to verify that LEGAL constrains the output signal of a latch to be underdefined and $\perp$ at time $t$, if and only if the input signal or clock signal of the latch is underdefined and $\perp$ at some time $t'$ that can make the value of the output signal at time $t$ ambiguous and thus underdefined.

Lemma 3.2 begins by demonstrating that the output signal of a latch cannot be valid if underdefined clock signal values make the output signal value ambiguous. Proof of the lemma is based on the observation that if $\phi$ has underdefined values, then there exists another clock signal $\phi'$, that represents the electrical signal that corresponds to $\phi$ more accurately. The lemma states that if $\text{LEGAL}(s_v, s_{v'}, \phi, t)$ is satisfied, and $s_v(t)$ is a valid value, then $\text{LEGAL}(s_v, s_{v'}, \phi', t)$ must also be satisfied, for any possible $\phi'$.

**Lemma 3.2** *Let $s_{v'}$ be the input signal, $s_v$ be the output signal and $\phi$ be the clock signal of an ideal latch, where $\text{LEGAL}(s_v, s_{v'}, \phi, \mathcal{T})$ for the latch is satisfied for all time $\mathcal{T}$. If $s_v(t) \neq \perp$, then for all digital clock signals $\phi'$, that equal $\phi$ for all $t'$ such that $\phi(t')$ is valid, $\text{LEGAL}(s_v, s_{v'}, \phi', t)$ for the latch is satisfied.*

**Proof:** If $\phi$ equals HIGH at time $t$, the lemma holds trivially, since $\phi'$ must also be HIGH at time $t$ and therefore $\text{LEGAL}(s_v, s_{v'}, \phi, t)$ and $\text{LEGAL}(s_v, s_{v'}, \phi', t)$ both require that $s_v(t)$ equals $s_{v'}$. In addition, none of the clauses of equation 3 ever reference any signal values for times not in the interval $(t_{\phi\text{-valid}}, t]$. Consequently, we can assume without loss of generality that $\phi$ is not equal to HIGH at time $t$, and $\phi$ and $\phi'$ are equal except over the interval $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]\!]$, where $t_{\phi\text{-invalid}}$ is in the interval if $t_{\phi\text{-invalid}}$ equals $t$.

To begin, consider the implications of $s_v(t) \neq \perp$. If $\phi$ is not HIGH at time $t$, and $\text{LEGAL}(s_v, s_{v'}, \phi, t)$ is satisfied for all time, then $s_v(t) \neq \perp$ implies that $s_v(t)$ equals $s_v(t_{\phi\text{-valid}})$ and $s_{v'}$ equals $s_v(t_{\phi\text{-valid}})$ over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]\!]$. In addition, since for all $\mathcal{T}$ in $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$ $\phi(\mathcal{T})$ equals $\perp$ and $\mathcal{T}_{\phi\text{-invalid}}$ equals $\mathcal{T}$, we can use equation 3 to conclude that $s_v$ also equals $s_v(t_{\phi\text{-valid}})$ over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]\!]$. In addition, if $\phi$ equals LOW at $t$, we observe that for all $\mathcal{T}$ in $[t_{\phi\text{-invalid}}, t]$, $\phi(\mathcal{T})$ equals LOW, and the interval $(\mathcal{T}_{\phi\text{-valid}}, \mathcal{T}_{\phi\text{-invalid}})$ is constant and equal to $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, and consequently we can once again use equation 3 to conclude that $s_v$ also equals $s_v(t_{\phi\text{-valid}})$ over $[t_{\phi\text{-invalid}}, t]$. Combining the preceding arguments, we conclude that if $s_v(t)$ is not $\perp$, then $s_v$ and $s_{v'}$ must both be stable and equal to $s_v(t_{\phi\text{-valid}})$ over the interval $(t_{\phi\text{-valid}}, t]$.

The remainder of the proof is divided into two parts. The first part shows that the lemma holds if $\phi'$ equals $\phi$ except over some closed subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]\!]$ where $\phi'$ is either HIGH or LOW over the entire subinterval. The second part uses the result of the first part to show the lemma for any $\phi'$ that equals $\phi$ except over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]\!]$.

Assume that $\phi'$ equals $\phi$ except over some closed subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]\!]$, $[t_s, t_e]$, where $\phi'$ is stable over the subinterval. If $\phi$ is LOW at time $t$, then $\phi'$ is also LOW at time $t$, and $\text{LEGAL}(s_v, s_{v'}, \phi', t)$ is satisfied if $s_v(t)$ equals $s_v(t_{\phi'\text{-valid}})$ and $s_{v'}(\mathcal{T}) = s_v(t_{\phi'\text{-valid}})$ for all $\mathcal{T} \in (t_{\phi'\text{-valid}}, t_{\phi'\text{-invalid}})$. Observe, however, that $(t_{\phi'\text{-valid}}, t_{\phi'\text{-invalid}})$ must be a subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, since $t_{\phi'\text{-valid}}$ equals $t_e$ and $t_{\phi'\text{-invalid}}$ equals $t_{\phi\text{-invalid}}$. Consequently, since $s_v$ and $s_{v'}$ must both be stable and equal to $s_v(t_{\phi\text{-valid}})$ over the interval $(t_{\phi\text{-valid}}, t]$, we

conclude that LEGAL$(s_v, s_{v'}, \phi', t)$ is satisfied. An identical argument applies when $\phi$ is $\perp$ at time $t$, and therefore we conclude that LEGAL$(s_v, s_{v'}, \phi', t)$ must be satisfied if $\phi'$ equals $\phi$, except over some closed subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$, $[t_s, t_e]$, where $\phi'$ is stable over the subinterval.

By repeating the preceding argument, we can conclude that the lemma holds for arbitrary closed interval differences between $\phi'$ and $\phi$ during $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$. Consequently, all that remains to be shown is that the lemma holds for any possible open interval differences between $\phi'$ and $\phi$, during $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$.

Open interval differences between $\phi'$ and $\phi$, during $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$, are in general not permissible, since $\phi'$ would no longer be a digital clock signal. The exceptions are the two intervals $(t_{\phi\text{-valid}}, t_s]$ and $[t_e, t_{\phi\text{-invalid}})$, where $t_s$ and $t_e$ are in the interval $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$. If $\phi'$ is stable with the appropriate values over these intervals, the effect would be to once again make the interval $(t_{\phi'\text{-valid}}, t_{\phi'\text{-invalid}}]$ a subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$. As before, however, since $s_v$ and $s_{v'}$ must both be stable and equal to $s_v(t_{\phi\text{-valid}})$ over the interval $(t_{\phi\text{-valid}}, t]$, we know that $s_v$ and $s_{v'}$ must both be stable and equal to $s_v(t_{\phi\text{-valid}})$ over the interval $(t_{\phi'\text{-valid}}, t]$, and therefore that LEGAL$(s_v, s_{v'}, \phi', t)$ must be satisfied. ∎

Unfortunately, lemma 3.2 only states that equation 3 does not wrongly require an output signal to be valid. Consequently, it is of limited usefulness, since it would, for example, trivially hold if LEGAL for a latch required that the output signal for the latch always be $\perp$. Observe, however, that the converse of lemma 3.2 is not true, since an input signal value of $\perp$ is another possible reason for the output signal of a latch to be invalid.

Lemma 3.3 is analogous to the converse of lemma 3.2, but takes into account the fact that invalidity of the input signal can affect the output signal. Specifically, it states that if the output signal of a latch is $\perp$, then some ambiguity introduced by an underdefined clock signal either itself introduces ambiguity about the value of the output signal, or allows ambiguity about the value of the input signal to be transferred to the output signal.

**Lemma 3.3** *Let $s_{v'}$ be the input signal, $s_v$ be the output signal and $\phi$ be the clock signal of an ideal latch, where* LEGAL$(s_v, s_{v'}, \phi, \mathcal{T})$ *for the latch is satisfied for all time $\mathcal{T}$. If $s_v(t) = \perp$, then there exist digital clock signals $\phi'$, $\phi'_1$ and $\phi'_2$, that equal $\phi$ for all $t'$ such that $\phi(t')$ is valid, and either*

1. LEGAL$(s'_v, s_{v'}, \phi'_1, t)$ *and* LEGAL$(s'_v, s_{v'}, \phi'_2, t)$ *cannot both be satisfied by any possible signal output signal $s'_v$ whose value at time $t$ is valid, or*

2. LEGAL$(s_v, s_{v'}, \phi', t)$ *constrains $s_v(t)$ to be the value of $s_{v'}(t')$ for some $t'$ less than or equal to $t$, where $s_{v'}(t')$ is $\perp$.*

**Proof:** The strategy for the proof is to examine each of the conditions that that allow $s_v(t)$ to equal $\perp$ when LEGAL$(s_v, s_{v'}, \phi', t)$ is satisfied. In each case, we show that at least one of the conditions stated in the lemma hold.

If $\phi$ equals HIGH at time $t$, the lemma holds trivially, since LEGAL$(s_v, s_{v'}, \phi, t)$ itself constrains $s_v(t)$ to be the value of $s_{v'}(t)$. In addition, none of the clauses of equation 3 ever reference any signal values for times not in the interval $(t_{\phi\text{-valid}}, t]$. Consequently, we can assume without loss of generality that $\phi$ is not equal to HIGH at time $t$, and $\phi'$, $\phi'_1$ and $\phi'_2$ are equal to $\phi$, except over the interval $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$, where $t_{\phi\text{-invalid}}$ is in the interval if $t_{\phi\text{-invalid}}$ equals $t$.

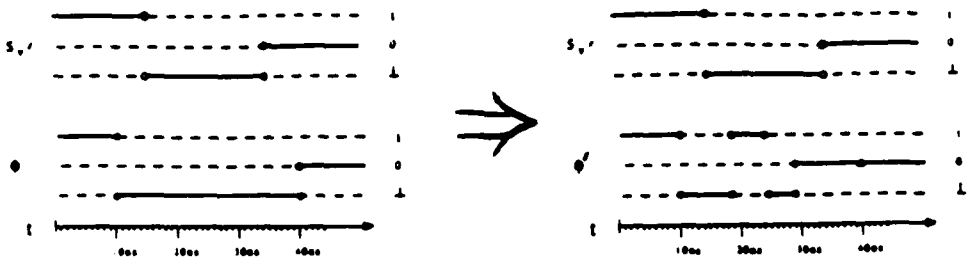

**Figure 8**: The figure illustrates how $\phi'$ from lemma 3.3 is constructed.

There are two cases to consider if $\phi$ equals LOW at time $t$. If $\phi$ equals LOW at time $t$ and LEGAL$(s_v, s_{v'}, \phi, t)$ is satisfied, $s_v(t)$ equal to $\perp$ implies that either $s_{v'}$ is not stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, or $s_{v'}$ is stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$ with a value not equal to $s_v(t_{\phi\text{-valid}})$.

If $\phi$ equals LOW at time $t$, and $s_{v'}$ is not stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, then $s_{v'}$ must have value $\perp$ over some non-zero length subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$. To see this, consider the following. If $s_{v'}$ is not stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, then either $s_{v'}$ is constant with value $\perp$ over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, or $s_{v'}$ holds two different values during $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$. In either case, however, $s_{v'}$ must have value $\perp$ over some non-zero length subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, since a digital signal must take on the value $\perp$ over some open interval between any two times that it has a valid value.

If $s_{v'}$ has value $\perp$ over some non-zero length subinterval of $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, we can construct a $\phi'$ such that $s_v(t)$ is equal to the value of $s_{v'}$ over the subinterval. We construct $\phi'$ by introducing into the subinterval a short closed interval, during which $\phi'$ is HIGH, followed by a short open interval, during which $\phi'$ is $\perp$, and setting $\phi'$ equal to LOW for times after the open $\perp$. Observe that $\phi'$ is such that the invalid value of $s_{v'}$ is latched and held through time $t$. We can thus conclude that the lemma holds when $\phi$ is LOW at time $t$, and $s_{v'}$ is not stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$. Figure 8 illustrates how $\phi'$ is constructed.

If $\phi$ equals LOW at time $t$, and $s_{v'}$ is stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$ with a value not equal to $s_v(t_{\phi\text{-valid}})$, we can show that $\phi$ must equal LOW at time $t_{\phi\text{-valid}}$. First, by the definition of digital signal and $t_{\phi\text{-valid}}$, $\phi$ cannot equal $\perp$ at $t_{\phi\text{-valid}}$, since this would imply that either STABLE$(\phi,$ LOW$)$ of STABLE$(\phi,$ HIGH$)$ contained a nonoverlapping open interval. Similarly, $\phi$ cannot equal HIGH at time $t_{\phi\text{-valid}}$, since $s_{v'}$ stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$ with a value not equal to $s_v(t_{\phi\text{-valid}})$ would imply that STABLE$(s_{v'}, x)$ would contain a nonoverlapping open interval, for some valid value $x$. Consequently, we can restrict the proof for $\phi$ equals LOW at time $t$, and $s_{v'}$ is stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$ with a value not equal to $s_v(t_{\phi\text{-valid}})$, to the case where $\phi$ must equal LOW at time $t_{\phi\text{-valid}}$.

If $\phi$ is equal to LOW at time $t_{\phi\text{-valid}}$, we can show the lemma by constructing an appropriate pair of clock signals, $\phi_1'$ and $\phi_2'$. The clock signal $\phi_1'$ simply has value LOW over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, so that $\phi_1'$ is LOW over $[t_{\phi\text{-valid}}, t]$. By lemma 3.1, $s_v$ at time $t$ must equal $s_v(t_{\phi\text{-valid}})$, if LEGAL$(s_v, s_{v'}, \phi_1', t)$ is to be satisfied. The clock signal $\phi_2'$ is constructed in a way similar to the clock signal $\phi'$ used when $s_{v'}$ is not stable over $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$. By construction, $s_v$ at time $t$ must equal the value of $s_{v'}$ during $(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}})$, if LEGAL$(s_v, s_{v'}, \phi_2', t)$ is to be satisfied. Since this case only applies if $s_{v'}$ is stable over

$(t_{\phi\text{-valid}}, t_{\phi\text{-invalid}}]$ with a value not equal to $s_v(t_{\phi\text{-valid}})$, we can conclude that LEGAL$(s'_v, s_{v'}, \phi'_1, t)$ and LEGAL$(s'_v, s_{v'}, \phi'_2, t)$ cannot both be satisfied by any possible signal $s'_v$ whose value at time $t$ is valid. Consequently, the lemma holds when $\phi$ is LOW at time $t$.

If $\phi$ is $\perp$ at time $t$, we can show the lemma by applying an argument identical to the one for $\phi$ equal to LOW at time $t$. Simply annotate $\phi'$, $\phi'_1$ and $\phi'_2$ so that they are all LOW at time $t$. ∎

Lemmas 3.4 and 3.5, verify facts about ideal latches that are needed in section 6 where we show that ideal circuit components can be used to model complete circuits. In addition, they verify the intuitive notion that the output signal of an ideal latch cannot suddenly become valid, while the clock signal of the latch is underdefined.

**Lemma 3.4** *Let $s_{v'}$ be the input signal, $s_v$ be the output signal and $\phi$ be the clock signal of an ideal latch, where* LEGAL$(s_v, s_{v'}, \phi, \mathcal{T})$ *for the latch is satisfied for all time $\mathcal{T}$. If $\phi$ is $\perp$ over the interval $(t_s, t_e]$, and for some $t \in (t_s, t_e]$ $s_{v'}$ is not stable over the interval $(t_s, t]$, then $s_v$ must be constant and $\perp$ over the interval $[t, t_e]$.*

**Proof:** The first step in the proof is simply to show that $s_{v'}$ not stable over the interval $(t_s, t]$ implies that $s_v(t)$ must equal $\perp$. To see this, consider the following. Since by definition $t_{\phi\text{-invalid}}$ must be less than or equal to $t_s$, we know that $(t_s, t]$ is a subinterval of $(t_{\phi\text{-valid}}, t]$ and consequently, that $s_{v'}$ is not stable over $(t_{\phi\text{-valid}}, t]$. Recalling the definition of stability, we therefore know that during $(t_{\phi\text{-valid}}, t]$ $s_{v'}$ is either 1) not constant or 2) constant with value $\perp$. In addition, since $\phi$ is $\perp$ at time $t$, the value of $s_v$ is determined in one of two ways. Either $s_{v'}(t') = s_v(t_{\phi\text{-valid}})$ for all $t' \in (t_{\phi\text{-valid}}, t]$ and $s_v(t) = s_v(t_{\phi\text{-valid}})$ or $s_{v'}(t') \neq s_v(t_{\phi\text{-valid}})$ for some time $t' \in (t_{\phi\text{-valid}}, t]$ and $s_v(t) = \perp$. Now, if $s_{v'}$ is not constant during $(t_{\phi\text{-valid}}, t]$, then for some time during $(t_s, t]$ its value cannot be equal to $s_v(t_{\phi\text{-valid}})$ and $s_v(t)$ must equal $\perp$. Alternately, if $s_{v'}$ is constant with value $\perp$ over $(t_{\phi\text{-valid}}, t]$, then either $s_v(t_{\phi\text{-valid}})$ equals $\perp$, or it does not. In either case, $s_v(t)$ must again equal $\perp$.

The finish the proof, we simply must show that $s_{v'}$ not stable over $(t_s, t]$ implies that $s_{v'}$ is not stable over $(t_s, \mathcal{T}]$, for all $\mathcal{T}$ in $[t, t_e]$. This is obviously true, however, since $(t_s, t]$ will always be a subinterval of $(t_s, \mathcal{T}]$. ∎

**Lemma 3.5** *Let $s_{v'}$ be the input signal, $s_v$ be the output signal and $\phi$ be the clock signal of an ideal latch, where* LEGAL$(s_v, s_{v'}, \phi, \mathcal{T})$ *for the latch is satisfied for all time $\mathcal{T}$. If $\phi$ is $\perp$ over some interval $(t_s, t_e]$, then $s_v$ can change value at most once during $(t_s, t_e]$, and the transition must be from a valid value to $\perp$.*

**Proof:** While this lemma may seem to be an obvious consequence of lemma 3.4, there are in fact four possible cases that must be considered in the proof. First, $s_{v'}(\mathcal{T})$ may equal $s_v(t_{\phi\text{-valid}})$ for all $\mathcal{T} \in (t_{\phi\text{-valid}}, t_e]$. In this case, $s_v$ equals $s_v(t_{\phi\text{-valid}})$ over $(t_s, t_e]$ and $s_v$ does not change value at all during the interval. Second, $s_{v'}(\mathcal{T})$ may be constant with some value that is not equal to $s_v(t_{\phi\text{-valid}})$ for all $\mathcal{T} \in (t_{\phi\text{-valid}}, t_e]$. In this case, $s_v$ equals $\perp$ over $(t_s, t_e]$ and once again $s_v$ does not change value at all during the interval. Third for some $t$ in $(t_s, t_e]$, $s_{v'}$ may be stable with value $s_v(t_{\phi\text{-valid}})$ over $(t_{\phi\text{-valid}}, \mathcal{T}]$ for all $\mathcal{T}$ in $(t_s, t]$, but not for any $\mathcal{T}$ in $(t, t_e]$. In this case, equation 3 and lemma 3.4, imply a single value transition, with $s_v$ equal to $s_v(t_{\phi\text{-valid}})$ over $(t_s, t']$, and equal to $\perp$ over $(t', t_e]$. Finally, for some $t$ in $(t_s, t_e]$, $s_{v'}$ may be stable with value not equal to $s_v(t_{\phi\text{-valid}})$ over $(t_{\phi\text{-valid}}, \mathcal{T}]$ for all $\mathcal{T}$ in

$(t_s, t]$, but not for any $\mathcal{T}$ in $(t, t_e)$. In this case, equation 3 and lemma 3.4 imply that the value of $s_v$ is $\perp$ over the entire interval, $(t_s, t_e)$, with no transitions. ∎

As a final note, we have assumed throughout this section, that the input signals of ideal components are digital signals, but have made no attempt to determine whether this implies that the output signals are digital. We address this question in section 6, in the more general context of complete circuits.

# 4 Strictly Clocked Circuits

This section presents our basic model for how circuit components are used to construct complete circuits. Our representations for a circuit and a computation on it are given, and we define the class of *strictly clocked circuits*, that is the subject of the remainder of this thesis.

A circuit is constructed by interconnecting a finite number of electrical components, such as transistors and logic gates. We represent a circuit as a directed graph $G = (V, E)$, where each vertex in $V$ is a circuit component, and $(u, v) \in E$ if the output signal of $u$ is an input signal of $v$. We assume that each component has an input edge for each of its input signals, which is equivalent to assuming that a circuit contains no "floating wires". Observe that in this representation, things like wires in a circuit are considered to be electrical components. Intuitively, this is how it should be, since objects like wires can in fact have very complex electrical behavior. A *computation* $C$ on a circuit $G = (V, E)$ is a set of signals, that contains for each component $v$ in $V$ a signal $s_v$.

A computation $C$ implicitly contains two sets of circuit inputs. The first is the set of clock signals for latches. We denote this set as $\Phi$, and assume that it contains only digital clock signals. The second is the set of signal values at time $-\infty$. This set is denoted as $Z$, and specifies the *initial conditions* of the circuit components. In this thesis, we consider $\Phi$ to be constant for all computations on a particular circuit, thus making $Z$ the only circuit input.

A *digital circuit* is a circuit $G = (V, E)$, where $V$ contains only digital circuit components, and computations on $G$ contain a unique signal $s_v$ for each component in $V$. Just as for individual digital components, we define a constraint for a digital circuit

$$\textsc{Legal}(C, t)$$

that if satisfied at time $t$, indicates that at time $t$ all signals in $C$ are consistent with the behavior of each digital component in $V$. An entire computation $C$ is said to be *legal*, if $\textsc{Legal}(C, \mathcal{T})$ is satisfied for all $\mathcal{T}$.

The stipulation that $s_v$ be unique is not equivalent to assuming that only a single component can "drive" a particular wire, since a buss can be viewed as an electrical component with an input for each component that can drive it. Observe, however, that the uniqueness of $s_v$ does imply that a signal in a computation only has to satisfy the output signal constraint of a single digital component. Consequently, our definition of a digital circuit includes a tacit assumption that the function associated with a functional element specifies a value for all possible combinations of valid input signal values. In the case of a buss, this assumption is equivalent to assuming that function associated with the buss is able to "resolve" any buss conflicts.

This thesis considers digital circuits that have all of the following properties:

1. The set of clocks $\Phi$ is finite, and its elements are fully specified digital clock signals. Circuits with this property are *statically clocked.*

2. There exists a *start time*, $t_{start}$ not equal to $-\infty$, such that all signals in $\Phi$ are constant over the interval $[-\infty, t_{start}]$. Circuits with this property are *statically initialized.*

3. For any time $t$, every cycle in $G$ contains at least one latch whose clock signal has value LOW at time $t$. Circuits with this property are *synchronous.*

A circuit with all these properties is said to be a *strictly clocked circuit.*

Of the three properties of a strictly clocked circuit, static initialization and synchroneity are effectively met by most MOS circuits. For example, most circuits incorporate some form of reset mechanism, that provides a means of establishing a start time for subsequent computations. Synchroneity is also generally met, since circuits typically use latches to prevent race-conditions between combinational logic blocks designed for minimum delay. Static initialization and synchroneity may not be met by designs that are pushing the state-of-the-art in circuit design, but we believe that with appropriate extensions our model can be generalized to circuits without these properties.

Many modern MOS circuits are not statically clocked, and it is less clear whether our model can be generalized to circuits without this property. The difficulty is related to the problem described in section 3, where it was noted that determining whether the output signals of functional elements are underdefined or undefined is too computationally expensive to be practical. If we are unable to determine the outputs of functional elements, using these outputs as clock signals introduces additional uncertainty. The implications of this uncertainty are at the time of this writing not clear.

# 5 Computational Predicates

In this section we define the class of *computational predicates*, that can be used to prove properties of strictly clocked circuits. We begin by discussing some of the issues of analyzing the time dependent behavior of a strictly clocked circuit, and then develop the mechanism of *computational ordering* for performing such analysis. In addition, the important concept of circuit *configurations* is also introduced.

A primary goal of this thesis is to show that our models for ideal circuit components also form a model for complete circuits. Until now, we have only examined the behavior of isolated digital components. It remains to be shown whether a group of digital components can be used in aggregate to form a circuit that exhibits reasonable behavior. For example, do legal computations exist for strictly clocked circuits?

Unfortunately, it is not clear how to prove properties of strictly clocked circuits, since there exists many indirect constraints on the signals in a legal computation. Consider, for example, the circuit in figure 9, where the latch $A$ not only places constraints between the values of signals $s_v$ and $s_{v'}$, but via functional element $B$ also indirectly places constraints between the values of $s_{v'}$ and $s_{v''}$. The situation is made even more difficult by the fact that the path via components $A$, $B$, $C$ and $D$ implies that there exists an indirect constraint between $s_v$ and itself. Accounting for all such indirect constraints becomes a formidable task, when considering an entire circuit for all possible values of time.

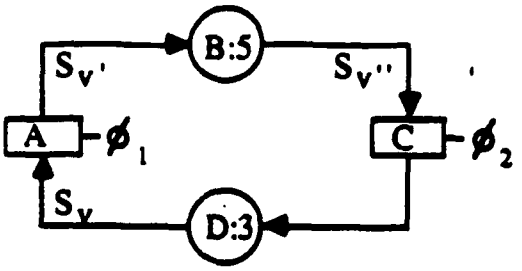


**Figure 9**: Shown is a four-component circuit. Observe that the latch $A$ not only places constraints between the values of signals $s_v$ and $s_{v'}$, but via functional element $B$ also indirectly places constraints between the values of $s_{v'}$ and $s_{v''}$

A common method for attacking problems that are characterized by indirect constraints, is induction. The method of induction that we will use, proves that a set of objects has a certain property by using the following three basic steps:

1. The desired property for the set is reformulated as a predicate on elements in the set. The predicate is such that if the predicate is not satisfied by any elements of the set, then the set has the original property.

2. An acyclic induction ordering of all elements in the set is established, such that if the predicate is satisfied by some element, then some element immediately earlier in the ordering must have also satisfied the predicate.

3. Elements that have no other elements before them in the ordering, are shown to not satisfy the predicate.

Once all three steps have been completed, the following reasoning is used to show that the original set must have the original property. Assume that there exist elements of the set that satisfy the predicate, and follow the acyclic induction ordering back to find an element $x$, such that $x$ satisfies the predicate, but all elements immediately before $x$ in the ordering do not satisfy the predicate. The only way for such an $x$ to exist, is if $x$ has no elements before it in the induction ordering. Consequently, since elements with no elements before them in the induction ordering have been explicitly shown to not satisfy the predicate, the assumption that any elements satisfied the predicate must have been in error.

The astute reader will note, however, that using inductive methods is much harder than the preceding description indicates. Particularly when continuous quantities, like time, are involved. The difficulty resides in the possibility that we could follow the ordering back indefinitely, without ever finding a suitable element $x$. Consider, for example, the property "is less than or equal to 5", for the non-negative real numbers, inductively ordered by the less-than operator. Clearly, the predicate "is less than or equal to 5" is not true at 0, which is the only non-negative real number that has no other non-negative reals that are less than it. In addition, for any number $x$ that is not less than or equal to 5, we can obtain another real number $y$ that is less than $x$, but still not less than or equal to 5, by using the equation $y = x - 0.5(x - 5)$. Consequently, the property "is less than or equal to 5", can be fitted

into our framework for an inductive proof, but it is obvious that not all non-negative real numbers are less than or equal to 5.

Fortunately, the *computational predicates* form a class of properties, for which inductive techniques can be adapted. A *predicate* is a function that maps signal-time pairs into the set {TRUE, FALSE}. A predicate $\mathcal{P}$ is said to be satisfied by a signal $s_v$ at time $t$, if $\mathcal{P}(s_v, t) = \text{TRUE}$. A predicate $\mathcal{P}$ is a *computational predicate* if it has the following five properties.

1. Every signal-time pair $(s, t)$ in a computation $C$ is mapped by $\mathcal{P}$ to the set {TRUE, FALSE}. A predicate with this property is *fully defined.*

2. $\mathcal{P}(s_v, t) = \text{TRUE}$ must imply that $\mathcal{P}(s_v, \mathcal{T}) = \text{TRUE}$ for all $\mathcal{T}$ greater than or equal to $t$. A predicate with this property is *monotone.*

3. If $s_v$ is the output signal for digital component $v$, then $\mathcal{P}(s_v, t) = \text{TRUE}$ must imply that $\mathcal{P}(s_{v'}, t') = \text{TRUE}$, for some $s_{v'}$ that is an input signal for $v$, and some $t'$ that is less than or equal to $t$. A predicate with this property is *causal.*

4. If $s_v$ is the output signal for some latch whose clock signal has value LOW at $t$, then $\mathcal{P}(s_v, \mathcal{T}) = \text{TRUE}$ over some interval $(t, t']$ must imply that $\mathcal{P}(s_v, t) = \text{TRUE}$, and $\mathcal{P}(s_v, \mathcal{T}) = \text{FALSE}$ over some interval $[t', t)$ must imply that $\mathcal{P}(s_v, t) = \text{FALSE}$. A predicate with this property is *latchable.*

5. If $s_v$ is the output signal for some latch whose clock signal has value LOW at $-\infty$, then $\mathcal{P}(s_v, -\infty) = \text{FALSE}$. A predicate with this property is *uninitial.*

The same induction ordering is used in all the inductive proofs for computational predicates. Let $C$ be a computation on a strictly clocked circuit, $G = (V, E)$. The ordering used to to inductively prove computational predicates is called the *computational ordering.* The computational ordering orders signals based on two orderings of signal-time pairs. The first ordering for signal-time pairs is based on the times, while the second ordering is based on the signals.

The first ordering for signal-time pairs is done *chronologically* by time. The signal-time pair $(s_v, t)$ satisfies the predicate $\mathcal{P}$ *chronologically before* the signal-time pair $(s_{v'}, t')$, if $\mathcal{P}(s_v, t)$ equals TRUE, and $t$ is less than $t'$. In general, chronological ordering is not sufficient to isolate a single signal-time pair that satisfies $\mathcal{P}$ before all others. For example, suppose that $\mathcal{P}(s_v, t)$ equals TRUE, if $s_v(t)$ equals $\perp$. We cannot guarantee that there exists a signal-time that satisfies $\mathcal{P}$ chronologically before all others, since equation 1 states that the input signal of a functional element changing to $\perp$ constrains the output signal of that functional element to change to $\perp$ simultaneously.

To order signal-time pairs that satisfy $\mathcal{P}$ at the same time $t$, we consider the structure of the circuit $G$. For any strictly clocked circuit $G$, and time $t$, the set of clock signals $\Phi$ maps to each latch a value for its clock input. Until some clock signal changes value, the circuit is equivalent in behavior to the circuit $G_t$ that is obtained by replacing latches mapped to HIGH with zero-delay identity functional elements, and deleting the input edges to latches mapped to LOW. The outputs of latches mapped to LOW are effectively external inputs, since their value cannot be effected by the values of other signals in the circuit. Since $G$ is strictly clocked, and therefore synchronous, $G_t$ will always be acyclic and consequently

provides a partial ordering of the digital components in $G$. The circuit $G_t$ is the *configuration* of $G$ at time $t$, and provides a way to break ties between signal-time pairs that satisfy $\mathcal{P}$ at time $t$. Let $s_v$ and $s_{v'}$ be the output signals of two digital components $v$ and $v'$ in $V$, respectively. The signal-time pair $(s_v, t)$ satisfies $\mathcal{P}$ *causally before* the signal time pair $(s_{v'}, t)$, if both $\mathcal{P}(s_v, t)$ and $\mathcal{P}(s_{v'}, t)$ equal TRUE, and there exists a path from $v$ to $v'$ in $G_t$.

Unfortunately, a naive combination of chronological and causal ordering does not allow us to inductively prove computational predicates. One might, for example, think that we could consider a signal-time pair $(s_v, t)$ to be before a signal-time pair $(s_{v'}, t')$, if either $(s_v, t)$ satisfies $\mathcal{P}$ chronologically before $(s_{v'}, t')$, or $(s_v, t)$ satisfies $\mathcal{P}$ causally before $(s_{v'}, t')$. Observe, however, that we can setup a situation identical to the earlier "less than or equal to 5" example, by defining a predicate that is TRUE for a signal-time pair $(s_v, t)$, if $t$ is greater than 5.

Fortunately, we can perform inductive proofs of computational predicates, by using chronological and causal ordering to establish a related ordering that is over the set of signals. Let $s_v$ and $s_{v'}$ be any two signals in a computation $C$. The signal $s_v$ is *computationally before* $s_{v'}$, for the predicate $\mathcal{P}$, if

1. there exists a time $t$, such that $(s_v, t)$ satisfies $\mathcal{P}$ chronologically before $(s_{v'}, \mathcal{T})$ for all $\mathcal{T}$, or

2. there exists an interval $[\![t_s, t_e]\!]$, such that for all $\mathcal{T}$ in $[\![t_s, t_e]\!]$,

   (a) $(s_v, \mathcal{T})$ and $(s_{v'}, \mathcal{T})$ satisfy $\mathcal{P}$,

   (b) $(s_v, t)$ and $(s_{v'}, t)$ do not satisfy $\mathcal{P}$ for any $t$ before $[\![t_s, t_e]\!]$, and

   (c) the configuration of the circuit $G_{\mathcal{T}}$ is constant, and $(s_v, \mathcal{T})$ satisfies $\mathcal{P}$ causally before $(s_{v'}, \mathcal{T})$.

Observe that computational ordering is an ordering of the set of signals. Since for any circuit/computation there are only a finite number of signals, computational ordering is not prone to the difficulties noted in the "less than or equal to 5" example.

Intuitively, computational ordering first orders signals by when they first satisfied $\mathcal{P}$, and breaks ties based on the configuration of the circuit at the time $\mathcal{P}$ was first satisfied. Extending our terminology, we say that signals ordered using chronological ordering are chronologically ordered, and similarly, signals ordered using causal ordering are causally ordered. Observe that since causal ordering for signal-time pairs is only a partial ordering, computational ordering of signals is also only partial. A signal $s_v$ is *computationally first* for a predicate $\mathcal{P}$, if no other signal is computationally before $s_v$.

The first step in showing that computational predicates can be proved inductively, is to show that if there exists a signal-time pair that satisfies a computational predicate $\mathcal{P}$, then there exists a computationally first signal for $\mathcal{P}$. Lemmas 5.1 and 5.2 prove the existence of a computationally first signal, in two steps. First, lemma 5.1 shows that chronological ordering, can isolate a set of signals, such that the interval $[\![t_s, t_e]\!]$ needed for applying causal ordering exists. Second, lemma 5.2 shows that at least one computationally first signal can be found in the set isolated by lemma 5.1.

**Lemma 5.1** *If $\mathcal{P}$ is a monotone predicate, then for any two digital signals $s_v$ and $s_{v'}$, either one signal is chronologically before the other or $\mathcal{P}(s_v, \mathcal{T})$ equals $\mathcal{P}(s_{v'}, \mathcal{T})$ for all $\mathcal{T}$.*

**Proof:** If $\mathcal{P}$ is monotone, then for any signal, the signal either satisfies $\mathcal{P}$ for all time, satisfies $\mathcal{P}$ for no time, or for some time $t$ does not satisfy $\mathcal{P}$ over the interval $[-\infty, t]$ but does satisfy $\mathcal{P}$ for all times after the interval. Observe, however, that if $\mathcal{P}(s_v, \mathcal{T})$ does not equal $\mathcal{P}(s_{v'}, \mathcal{T})$ for all $\mathcal{T}$, then $\mathcal{P}$ must be satisfied by one signal at some time strictly before all the times that $\mathcal{P}$ is satisfied by the other, which by definition implies that one signal is chronologically before the other. ∎

We now show that if a computational predicate $\mathcal{P}$ is ever satisfied by some signal-time pair from a computation on a strictly clocked circuit, then there exists some signal in the computation that is computationally first for $\mathcal{P}$.

**Lemma 5.2** *Let $\mathcal{P}$ be some computational predicate, and $C$ be a computation on some strictly clocked circuit $G$. If for some signal $s_v$ in $C$ and time $t$, $\mathcal{P}(s_v, t)$ is* TRUE, *then there exists a signal in $C$ that is computationally first for $\mathcal{P}$.*

**Proof:** By definition, there exists a computationally first signal, unless there exists some set of signals $S$ such that for every signal $s_v$ in $S$, there exists another signal $s_{v'}$ in $S$ that is computationally before $s_v$. We can show, however, that such a set of signals cannot exist.

We first show that the existence of such a set $S$ implies that there must exist a time $t$, such that all signals in the sequence do not satisfy $\mathcal{P}$ over the interval $[-\infty, t]$ but do satisfy $\mathcal{P}$ for all times after the interval. Consider the following argument. Since there are only a finite number of signals, the existence of $S$ implies that there exists a cyclic sequence of signals, such that each signal in the sequence is computationally before the next. If we consider each adjacent pair of signals in the sequence, lemma 5.1 guarantees that any pair must either be chronologically ordered or satisfy $\mathcal{P}$ at all the same times. Observe, however, that no pair of adjacent signals can be chronologically ordered, since we could then follow the cyclic sequence back and conclude that some signal was chronologically before itself. Thus, the existence of $S$ implies that there must exist a time $t$, such that all signals in the sequence do not satisfy $\mathcal{P}$ over the interval $[-\infty, t]$ but do satisfy $\mathcal{P}$ for all times after the interval.

Now, since $G$ is strictly clocked, there must exist a time $t' \geq t$ such that the configuration of $G$ is constant over the interval $[t, t']$, where $[t, t']$ contains $t$ if and only if the interval $[-\infty, t]$ did not. To see this, consider the following. Since $G$ is strictly clocked, and therefore statically clocked, $\Phi$ contains a finite number of locally finite digital clock signals. Consequently, $G$ can change configurations only a finite number of times during any interval $[t_1, t_2]$, where $t_1, t_2 \in \mathbb{R}$. In addition, since $G$ is strictly clocked, and therefore statically initialized, we can also conclude that $G$ can change configurations only a finite number of times during any interval $[t_1, t_2]$, where $t_1, t_2 \in \mathbb{R} \cup \{-\infty\}$. Now, if we let $t''$ be any time greater than $t$, by the preceding arguments, $G$ can change configuration only a finite number of times during the interval $[t, t'']$. Consequently, there must exist a $t'$ greater than or equal to $t$ but less than $t''$ such that the configuration of $G$ is constant over $[t, t']$.

Now, let $G_t$ be the configuration corresponding to the interval $[t, t']$. Since none of the signals in the sequence are chronologically before any of the others, the fact that each signal is computationally before the next, implies that there exists a cycle in $G_t$. This, however, is not possible, since $G$ is strictly clocked, and therefore synchronous. ∎

We can now prove the theorem that will allow us to easily determine whether a computation on a strictly clocked circuit has a particular property. Theorem 5.1 states that no computational predicate can be satisfied by any signal-time pair from a computation on a strictly clocked circuit. Consequently, a computation on a strictly clocked circuit can be easily shown to have a particular property, if the negation of the property can be cast as a computational predicate.

**Theorem 5.1** *No computational predicate can be satisfied by any signal-time pair from a computation on a strictly clocked circuit.*

**Proof:** Let $\mathcal{P}$ be a computational predicate, and $C$ be a computation on a strictly clocked circuit $G$. If $\mathcal{P}$ is satisfied by some signal-time pair from $C$, then lemma 5.2 implies that there exists in $C$ a computationally first signal for $\mathcal{P}$. We can show, however, that no such computationally first signal can exist, and thus conclude that $\mathcal{P}$ cannot have been satisfied by any signal-time pair from $C$.

Let $s_v$ be the implied computationally first signal. Since $\mathcal{P}$ is computational, and therefore uninitial and monotone, we know that there exists a $t$ such that $\mathcal{P}$ is not satisfied by $s_v$ over the interval $[-\infty, t)\!]$, but $\mathcal{P}$ is satisfied by $s_v$ for all time after the interval. In addition, by repeating the argument used in lemma 5.2 we know that there exist a time $t' \geq t$ such that the configuration of $G$ is constant over the interval $(\![t, t']$, where $(\![t, t']$ contains $t$ if and only if the interval $[-\infty, t)\!]$ did not.

Using the interval $(\![t, t']$, we can show that $s_v$ must be the output signal of a latch whose clock signal has value LOW at $t$. Consider the following argument. If $s_v$ were the output signal of anything other than a latch whose clock signal had value LOW over $(\![t, t']$, then the fact that $\mathcal{P}$ is computational, and therefore causal would directly imply that $s_v$ was computationally preceded by some other signal. Consequently, $s_v$ must be the output signal of a latch whose clock signal has value LOW over $(\![t, t']$. In addition, since $G$ is strictly clocked, and therefore statically clocked, the clock signal for the latch that corresponds to $s_v$ must be digital and must have value LOW over the closed interval $[t, t']$. Consequently, $s_v$ must be the output signal of a latch whose clock signal has value LOW at $t$.

Proof of the theorem is now easy. So far, we have demonstrated that if $\mathcal{P}$ is satisfied by some signal-time pair from $C$, then there must exist a signal $s_v$ in $C$ such that

1. $s_v$ is the output signal of a latch whose clock signal has value LOW at time $t$,
2. $\mathcal{P}$ is not satisfied by $s_v$ over the interval $[-\infty, t)\!]$, and
3. $\mathcal{P}$ is satisfied by $s_v$ for all time after the interval $[-\infty, t)\!]$.

No such signal can exist, however, since the fact that $\mathcal{P}$ is latchable implies conflicting values for $\mathcal{P}(s_v, t)$. ∎

Theorem 5.1 is significant for three reasons. First, it provides a way to perform a "generic" inductive proof for a property, by simply casting the property as a predicate, showing the negation of the predicate to be computational, and invoking theorem 5.1. In effect, the reference to theorem 5.1 repeats the basic inductive proof that was amortized across the proofs of lemma 5.1, lemma 5.2 and theorem 5.1. This ability to indirectly repeat the inductive argument represents a great notational convenience. Second, the

theorem provides a mechanism for integrating the limit arguments that will generally be needed to show the latchability of a predicate into the inductive framework of computational ordering. Third, the theorem allows subsequent proofs to focus on the significant properties of a predicate, rather than the arguments that make use of them.

# 6 Model Consistency

In this section, we formally confirm the ability of digital components to model complete circuits. Specifically, we show that there exists for any strictly clocked circuit $G$ and set of initial conditions for $G$, a unique digital computation $C$ for which $\textsc{Legal}(C, \mathcal{T})$ is satisfied for all $\mathcal{T}$. In addition, the usefulness and generality of theorem 5.1 is demonstrated by the use of computational predicates to prove the different properties of $C$.

Electrical circuits possess three *natural* properties, that strictly clocked circuits must share if strictly clocked circuits are to be considered representations for electrical circuits. First, for any set of initial conditions, an electrical circuit has some behavior, whether it be to compute some function or to catch fire. Second, the behavior of an electrical circuit is such that the output signals of components are meaningful input signals for the components that use them for this purpose. Third, the behavior of an electrical circuit is deterministic and therefore unique for a particular set of initial conditions. Circuits that possess all these properties is said to be *well formed.* If strictly clocked circuits are not well formed, then strictly clocked circuits are incapable of accurately reflecting our most basic intuitive notions about electrical circuits.

Of the three natural properties, the most important from a formal stand point is the existence of some behavior. For strictly clocked circuits, this property is equivalent to the assertion that for any initial conditions, there exists a computation $C$ that satisfies $\textsc{Legal}(C, \mathcal{T})$ for all $\mathcal{T}$. If no such computation exists, then we are left without the ability to draw conclusions based on equations 1 and 3, and in effect are left with nothing but a meaningless formalism. Observe that this property does not assert that the signals in $C$ must necessarily be digital, just as the corresponding property for electrical circuits allows for catastrophic behavior such as catching fire.

Once the existence of a computation is established, the other two natural properties become meaningful. The property that the output signals of components be meaningful inputs signals, is equivalent to stating that $C$ contains only digital signals. Similarly, the property that behavior is deterministic, is equivalent to stating that for any set of initial conditions, $C$ is unique. Both these properties are "nice" properties, that we expect properly operating circuits to possess. Observe, however, that even without these properties, it would still be possible to use equations 1 and 3 to reason about $C$.

The following four lemmas state that strictly clocked circuits possess all the properties for being well formed. In addition, the proofs for the lemmas demonstrate how computational predicates can be applied to different types of properties. Lemmas 6.1 and 6.2 show that if a legal computation exists for a strictly clocked circuit, then the computation must contain only digital signals. Lemma 6.3 and theorem 6.1 show that there exists a unique legal computation for any strictly clocked circuit and set of initial conditions. Lemmas 6.1 and 6.2 are presented first, because the straight forward computational predicates they utilize provide an easier introduction to how properties can be cast as computational predicates.

**Lemma 6.1** *Let $C$ be a computation on some strictly clocked circuit $G = (V, E)$. If $C$ satisfies* LEGAL *for all time, then for each signal $s_v$ in $C$, and value $x \in \mathcal{D}$, the values $t$, such that $s_v(t) = x$, form a set* STABLE$(s_v, x)$ *of nonoverlapping closed intervals.*

**Proof:** The lemma obviously holds, if we can show that no signal-time pairs from $C$ satisfy the predicate $\mathcal{P}_1$, where:

> **Predicate.** $\mathcal{P}_1(s_v, t)$ equals TRUE if, for some $x \in \mathcal{D}$, the set STABLE$(s_v, x) \cap \{t' : t' \leq t\}$ contains a nonoverlapping open interval, and FALSE otherwise.

Thus, we can prove the lemma by showing that $\mathcal{P}_1$ is computational, and invoking theorem 5.1.

As will be typical, $\mathcal{P}_1$ is clearly fully defined, monotone and uninitial, and can be shown to be causal and latchable with an analysis of the different clauses of equations 1 and 3. The overall strategy for causality is to assume that $\mathcal{P}_1(s_v, t) =$ TRUE, and demonstrate on a clause by clause basis that if equations 1 and 3 are satisfied, then some input signal $s_{v'}$ must also have a nonoverlapping open interval in STABLE$(s_{v'}, x) \cap \{t' : t' \leq t\}$, for some value $x$. Specifically, we let $(\!|t_1, t_2|\!)$ be the nonoverlapping open interval implied by $\mathcal{P}_1(s_v, t) =$ TRUE and consider how each clause in equations 1 and 3, that may have to be satisfied by $s_v$ over $(\!|t_1, t_2|\!)$, effects the value of $s_v$ at the end points $t_1$ and $t_2$. In each case, we can show that either the end point must be included in $(\!|t_1, t_2|\!)$, and consequently cannot have caused $\mathcal{P}_1$ to be TRUE, or $\mathcal{P}_1(s_{v'}, t_2) =$ TRUE, for some input signal $s_{v'}$. This is sufficient for showing $\mathcal{P}_1$ to be causal, since $t_2$ must be less than or equal to $t$. Since equations 1 and 3 have a total of six different clauses, we adopt the convention of itemizing parts of an analysis by the type of ideal element involved and the specific case being considered.

### Functional Elements

$s_v(t) = f(s_1(t), s_2(t), \ldots, s_k(t))$:

This case applies if $s_i$ is stable for all $i = 1, 2, \ldots, k$ over the interval $[t - d, t]$. Consequently, the inclusion of $(\!|t_1, t_2|\!)$ in STABLE$(s_v, x) \cup \{t' : t' \leq t\}$ implies that all $s_i$ are stable over the interval $(\!|t_1 - d, t_2|\!)$. Observe, however, that if all input signals were stable over $[t_1 - d, t_2]$ then $s_v$ would have to be stable over $[t_1, t_2]$ and contradicts the assumption that $(\!|t_1, t_2|\!)$ was open. Consequently, there must exist some $s_i$ that is stable over the interval $(\!|t_1, t_2|\!)$ but not stable over the interval $[t_1 - d, t_2]$. This directly implies that for the value $x'$, that $s_i$ has during the interval $(\!|t_1, t_2|\!)$, the set STABLE$(s_i, x')$ must contain an interval that is open at either $t_1$ or $t_2$. Thus, since $t_1$ must be less than or equal to $t_2$, $\mathcal{P}_1(s_i, t_2)$ must equal TRUE.

$s_v(t) = \perp$:

Since STABLE$(s_v, x)$ is defined only for elements of the data domain $\mathcal{D}$, $s_v$ cannot be $\perp$ at any time during $(\!|t_1, t_2|\!)$. Consequently, this case is not applicable.

### Latches

For simplicity, this part of the analysis will not be based on $(\!|t_1, t_2|\!)$, but rather on two subintervals of $(\!|t_1, t_2|\!)$. Since $G$ is strictly clocked, and therefore statically clocked and initialized, $G$ can take on only a finite number of configurations during the interval $[-\infty, t_2]$. Consequently, there must exist $t_1' \geq t_1$ and $t_2' \leq t_2$, such that the configuration of $G$ is constant over the intervals $(\!|t_1, t_2'|]$ and $[t_1', t_2|\!)$. By basing our analysis on these two *end* intervals, we avoid the complications introduced by transitions of the clock input $\phi$, and allow ourselves to consider each end point separately. This is important since, for any given circuit, it is likely that the different ends of $(\!|t_1, t_2|\!)$ will be covered by different clauses.

$s_v(t) = s_{v'}(t)$ if $\phi(t) =$ HIGH:

Showing causality is simple for this case, given the assumption that $\phi$ is a digital clock signal. Consider the end interval $(\!|t_1, t_2'|]$. If $(\!|t_1, t_2'|]$ is closed, the end interval can be ignored. If it is open, consider the following. Since $\phi$ is a digital clock signal, STABLE($\phi$, HIGH) must contain only closed intervals, and thus, $\phi$ must have value HIGH, and $s_v$ must equal $s_{v'}$, during the entire interval $[t_1, t_2']$. Consequently, $s_v$ stable, but not equal to $s_v(t_1)$, during the interval $(\!|t_1, t_2'|]$ directly implies $s_{v'}$ stable, but not equal to $s_{v'}(t_1)$, during the interval $(\!|t_1, t_2'|]$. The other end interval $[t_1', t_2)\!|$ can be covered with an identical argument.

$s_v(t) = s_v(t_{\phi\text{-valid}})$ if $\phi(t) = \perp$:

For $(\!|t_1, t_2'|]$, this clause always implies that $t_1$ is in $(\!|t_1, t_2'|]$. To see this, observe that since $s_v$ is assumed to be stable over $(\!|t_1, t_2'|]$, it cannot be equal to $\perp$ over that interval, and thus its value during $(t'_{2\phi\text{-valid}}, t_2']$ must equal its value at $t'_{2\phi\text{-valid}}$. Consequently $s_v$ must be stable over the interval $[t'_{2\phi\text{-valid}}, t_2']$ and $(\!|t_1, t_2'|]$ cannot have been open, since $t'_{2\phi\text{-valid}}$ must be less than or equal to $t_1$.

For $[t_1', t_2)\!|$, we must consider the value of $\phi$ at $t_2$. If $\phi(t_2)$ is equal to LOW, then $s_v(t_2)$ will equal $s_v(t_{2\phi\text{-valid}})$ if $s_{v'}$ equals $s_v(t_{2\phi\text{-valid}})$ over $(t_{2\phi\text{-valid}}, t_{2\phi\text{-invalid}})$. Consequently, since $s_v$ stable over $[t_1', t_2)\!|$ implies that $s_{v'}$ must be stable over $(t_{2\phi\text{-valid}}, t_2)$, and $t_{2\phi\text{-invalid}}$ equals $t_2$, $s_v$ at time $t_2$ must be equal to its value over $[t_1', t_2)$ and thus by definition $s_v$ must be stable over $[t_1', t_2]$.

If $\phi(t_2)$ equals $\perp$, then $[t_1', t_2)\!|$ can be open only if $s_{v'}$ is stable over $(t_{2\phi\text{-valid}}, t_2)$, but not over $(t_{2\phi\text{-valid}}, t_2]$. This, however, directly implies that STABLE$(s_{v'}, x) \cap \{t' : t' \leq t\}$, contains an open interval for the value of $s_{v'}$ during $(t_{2\phi\text{-invalid}}, t_2)$.

Finally if $\phi(t_2)$ equals HIGH, then $s_v(t_2)$ is equal to $s_{v'}(t_2)$. Consequently, since $s_v$ stable over $[t_1', t_2)\!|$ implies that $s_{v'}$ must be stable over $(t_{2\phi\text{-valid}}, t_2)$, $[t_1', t_2)\!|$ can be open only if $s_{v'}$ at time $t_2$ is not equal to its value over the interval $(t_{2\phi\text{-invalid}}, t_2)$. Just as for $\phi(t_2)$ equals $\perp$, however, this implies that STABLE$(s_{v'}, x) \cap \{t' : t' \leq t\}$, contains an open interval for the value of $s_{v'}$ during $(t_{2\phi\text{-invalid}}, t_2)$.

$s_v(t) = s_v(t_{\phi\text{-high}})$ if $\phi(t) =$ LOW:

For $(\!|t_1, t_2'|]$, the fact that $\phi$ is a digital clock signal, implies that $\phi$ must be LOW over $[t_1, t_2']$. Lemma 3.1 therefore implies that $s_v$ must be stable over $[t_1, t_2']$ and consequently, that $(\!|t_1, t_2'|]$ must be closed. The second end interval $[t_1', t_2)\!|$, can be covered with an identical argument.

$s_v(t) = \perp$:

Just as for functional elements, this case is not applicable, since STABLE$(s_v, x)$ is defined only for elements of the data domain $\mathcal{D}$. Having addressed all clauses of equations 1 and 3, we can conclude that $\mathcal{P}_1$ is causal.

To demonstrate that $\mathcal{P}_1$ is latchable, we assume that $\mathcal{P}_1$ is not latchable and show that this results in a contradiction. First, we let $s_v$ be the output signal of some latch whose clock signal is LOW at time $t$. Now to prove that if $\mathcal{P}_1(s_v, \mathcal{T}) =$ FALSE over $[t', t)$ then $\mathcal{P}_1(s_v, t) =$ FALSE, we show that if $\mathcal{P}_1(s_v, \mathcal{T}) =$ FALSE over $[t', t)$ does not imply that $\mathcal{P}_1(s_v, t) =$ FALSE, then there must exist an open interval $[t_s, t)$ such that the configuration of the circuit is constant over $[t_s, t)$, and the value of $s_v$ over $[t_s, t)$ is constant and not equal to the value of $s_v$ at $t$. The existence of such a interval will be shown to be a contradiction, since such an interval is not possible for any value of $\phi$ over $[t_s, t)$. The proof that if $\mathcal{P}_1(s_v, \mathcal{T}) =$ TRUE over $(t, t']$ then $\mathcal{P}_1(s_v, t) =$ TRUE, is completely symmetrical.

Let $s_v$ be the output signal of some latch whose clock signal is LOW at time $t$, and assume that $\mathcal{P}_1(s_v, \mathcal{T}) =$ FALSE over some interval $[t', t)$ but $\mathcal{P}_1(s_v, t) =$ TRUE. Since $\mathcal{P}_1$ for $s_v$is not TRUE for any time less than $t$, the open interval implied by $\mathcal{P}_1(s_v, t) =$ TRUE must in fact be open at $t$. Consequently, by repeating the reasoning from the proof of lemma 5.2, we can conclude that $\mathcal{P}_1(s_v, \mathcal{T}) =$ FALSE over some interval $[t', t)$ but $\mathcal{P}_1(s_v, t) =$ TRUE, implies that there exists some $t_s$ less than $t$, such that the configuration of the circuit is constant over $[t_s, t)$, and the value of $s_v$ over $[t_s, t)$ is constant and not equal to the value of $s_v$ at $t$. We can show, however, that such a $t_s$ cannot exist for any value of $\phi$ over the interval $[t_s, t)$.

It is easy to show that $\phi$ cannot equal LOW or HIGH over $[t_s, t)$. If $\phi$ is LOW over $[t_s, t)$, then the fact that $\phi$ is assumed to be LOW at time $t$ implies that $\phi$ is LOW over $[t_s, t]$ and consequently that, by lemma 3.1, the value of $s_v$ over the interval $[t_s, t)$ cannot be different from the value of $s_v$ at $t$. In addition, $\phi$ cannot be HIGH over $[t_s, t)$, since $\phi$ equal to LOW at time $t$ would imply that $\phi$ was not a digital signal.

Showing that $\phi$ cannot be $\perp$ over $[t_s, t)$, is more intricate, since it requires examining two possible cases. First, if $s_v$ has a valid value over $[t_s, t)$, then it must have value $s_v(t_{\phi\text{-valid}})$ over $[t_s, t)$. Consequently, we can conclude that the input signal $s_{v'}$ of the latch is constant and equal to $s_v(t_{\phi\text{-valid}})$ over $[t_{\phi\text{-valid}}, t)$. Observe, however, that equation 3 then directly implies that the value of $s_v$ at $t$ must also equal $s_v(t_{\phi\text{-valid}})$. Similarly, if $s_v$ does not have a valid value over $[t_s, t)$, then it must have value $\perp$ over $[t_s, t)$, and the input signal $s_{v'}$ of the latch must not be equal to $s_v(t_{\phi\text{-valid}})$ over $[t_{\phi\text{-valid}}, t)$. As before, however, equation 3 then directly implies that the value of $s_v$ at $t$ must also equal $\perp$. Consequently, if $\phi$ is $\perp$ over $[t_s, t)$, then the value of $s_v$ over the interval $[t_s, t)$ cannot be different from the value of $s_v$ at $t$.

The proof that, if $\mathcal{P}_1(s_v, \mathcal{T}) =$ TRUE over $(t, t']$ then $\mathcal{P}_1(s_v, t) =$ TRUE, is symmetrical, except that we establish the existence of a $t_e$ greater than $t$, such that the configuration of the circuit is constant over $(t, t_e]$, and the value of $s_v$ over $(t, t_e]$ is constant and not equal to the value of $s_v$ at $t$. It is easy to show, that such a $t_e$ cannot exist for any value of $\phi$ over the interval $(t, t_e]$. ∎

The second property of a digital signal requires that for any signal $s_v$ and value $x \in \mathcal{D}$, STABLE$(s_v, x)$ is locally finite. Lemma 6.2 shows that STABLE$(s_v, x)$ is locally finite for any signal $s_v$ in a legal computation.

**Lemma 6.2** *Let $C$ be a computation on some strictly clocked circuit $G = (V, E)$. If $C$ satisfies* LEGAL *for all time, then for each signal $s_v$ in $C$, and value $x \in \mathcal{D}$, the values $t$, such that $s_v(t) = x$, form a set* STABLE$(s_v, x)$ *that is locally finite.*

**Proof:** The lemma obviously holds, if we can show that no signal-time pairs from $C$ satisfy the predicate $\mathcal{P}_2$, where:

> **Predicate.** $\mathcal{P}_2(s_v, t)$ equals TRUE, if $s_v$ changes value an infinite number of times during the interval $[-\infty, t]$, and FALSE otherwise.

Thus, we can prove the lemma by showing that $\mathcal{P}_2$ is computational, and invoking theorem 5.1.

Just as for $\mathcal{P}_1$, the predicate $\mathcal{P}_2$ is clearly fully defined, monotone and uninitial, and can be shown to be causal and latchable with an analysis of the different clauses of equations 1 and 3. The strategy for causality is to assume that $\mathcal{P}_2(s_v, t) =$ TRUE, and demonstrate on a clause by clause basis that if equations 1 and 3 are satisfied, then some input signal $s_{v'}$ for the component that $s_v$ corresponds to must also change value an infinite number of times during $[-\infty, t]$.

### Functional Elements

For functional elements, we show that if all input signals change value only a finite number of times during $[-\infty, t]$, then $s_v$ can changed values only a finite number of times during $[-\infty, t]$. This is sufficient for showing causality, since it implies that if $s_v$ changes value an infinite number of times during the interval $[-\infty, t]$, then some input signal must also.

If all input signals for $v$ change value only a finite number of times during $[-\infty, t]$, then there exists a partition $\pi$ of $[-\infty, t]$ that partitions $[-\infty, t]$ into a finite number of intervals, where all input signals, for the component that $s_v$ corresponds to, are constant during each interval. This is an obvious consequence of the fact there are only a finite number of inputs to any functional element, and therefore only a finite total number of value transitions, across all input signals.

It is now easy to see that a finite number of value transitions for all inputs during $[-\infty, t]$ must imply that $s_v$ cannot change value an infinite number of times during $[-\infty, t]$. Given the existence of the partition $\pi$, an infinite number of value transition of $s_v$ would imply that $s_v$ must change value an infinite number of times during some interval $[\![t_s, t_e]\!]$ where all inputs are constant during $[\![t_s, t_e]\!]$. It is obvious from equation 1, however, that the only value change that $s_v$ can under go, during any interval over which all its inputs are constant, is from $\perp$ to some valid value, and consequently, that $s_v$ can change value at most once during $[\![t_s, t_e]\!]$.

### Latches

The argument for level-sensitive latches is identical to that for functional elements, except that we use the clock signal $\phi$ to isolate a finite interval during which $s_v$ must change value an infinite number of times.

Since $G$ is strictly clocked, and therefore statically clocked and initialized, $\phi$ can change value at most a finite number of times during $[-\infty, t]$. To see this, consider the following.

Since $G$ is statically initialized, there must exist a time $t_{start}$ not equal to $-\infty$, such that $\phi$ is constant over the interval $[-\infty, t_{start}]$. Consequently, all transitions in the value of $\phi$ must occur during the interval $[t_{start}, t]$. Since $t_{start}$ is not equal to $-\infty$, however, the locally finite property of $\phi$ guarantees that $\phi$ cannot change more that a finite number of times during $[t_{start}, t]$.

Just as for functional elements, it is now easy to see that a finite number of value transitions of the input signal of the latch $s_{v'}$, during $[-\infty, t]$, implies that $s_v$ cannot change value an infinite number of times during $[-\infty, t]$. An infinite number of value transition of $s_v$ would imply that $s_v$ must change value an infinite number of times during some interval $(\!|t_s, t_e|\!)$ where $\phi$ is constant during $(\!|t_s, t_e|\!)$. Observe, however, that if $\phi$ is LOW over $(\!|t_s, t_e|\!)$, then by lemma 3.1 $s_v$ cannot change value at all during $(\!|t_s, t_e|\!)$. Alternately, if $\phi$ is HIGH, $s_v$ must equal $s_{v'}$ over $(\!|t_s, t_e|\!)$, and $s_v$ cannot change values an infinite number of times during $(\!|t_s, t_e|\!)$, since $s_{v'}$ is assumed to change value only a finite number of times during $(\!|t_s, t_e|\!)$. Finally, if $\phi$ is $\perp$ over $(\!|t_s, t_e|\!)$, lemma 3.5 implies that $s_v$ can change value at most once during $(\!|t_s, t_e|\!)$. Since HIGH, LOW and $\perp$ are the only possible values for $\phi$, we can conclude that a finite number of value transitions of $s_{v'}$, during $[-\infty, t]$, must imply that $s_v$ cannot change value an infinite number of times during $[-\infty, t]$.

The predicate $\mathcal{P}_2$ can be shown to be latchable with an argument completely analogous to that used for $\mathcal{P}_1$. First, we let $s_v$ be the output signal of some latch whose clock signal is LOW at time $t$. To prove that if $\mathcal{P}_2(s_v, \mathcal{T}) =$ FALSE over some interval $[t', t)$ then $\mathcal{P}_2(s_v, t) =$ FALSE, we show that if $\mathcal{P}_2(s_v, \mathcal{T}) =$ FALSE over $[t', t)$ does not imply that $\mathcal{P}_2(s_v, t) =$ FALSE, then there must exist an open interval $[t_s, t)$ such that the configuration of the circuit is constant over $[t_s, t)$, and $s_v$ changes value an infinite number of times during $[t_s, t)$. The existence of such a interval will be shown to be a contradiction, since such an interval is not possible for any value of $\phi$ over $[t_s, t)$. The proof that, if $\mathcal{P}_2(s_v, \mathcal{T}) =$ TRUE over $(t, t']$ then $\mathcal{P}_2(s_v, t) =$ TRUE, is completely symmetrical.

Let $s_v$ be the output signal of some latch whose clock signal is LOW at time $t$, and assume that $\mathcal{P}_2(s_v, \mathcal{T}) =$ FALSE over $[t', t)$ but $\mathcal{P}_2(s_v, t) =$ TRUE. Since $\mathcal{P}_2(s_v, \mathcal{T})$ is not TRUE for any time $\mathcal{T}$ less than $t$, but is TRUE for $t$, $s_v$ must change value an infinite number of times during the interval $[t'', t)$, where $t''$ is any time in $[t', t)$. Consequently, by repeating the reasoning from the proof of lemma 5.2, we know that there must exist some $t_s$ less than $t$, such that the configuration of the circuit is constant over $[t_s, t)$, and $s_v$ must change value an infinite number of times during the interval $[t_s, t)$. We can show, however, that such a $t_s$ cannot exist for any value of $\phi$ over the interval $[t_s, t)$.

It is easy to show that $\phi$ cannot equal LOW, HIGH or $\perp$, over $[t_s, t)$. If $\phi$ is LOW over $[t_s, t)$, then $s_v$ cannot change value an infinite number of times during $[t_s, t)$, since by lemma 3.1 $s_v$ cannot change value at all during $[t_s, t)$. Alternately, $\phi$ cannot be HIGH over $[t_s, t)$, since the assumption that $\phi$ equals LOW at time $t$ would then imply that $\phi$ was not a digital clock signal. Finally, If $\phi$ is $\perp$ over $[t_s, t)$, then $s_v$ cannot change value an infinite number of times during $[t_s, t)$, since by lemma 3.5 $s_v$ can change value at most once during $[t_s, t)$.

The proof that if $\mathcal{P}_2(s_v, \mathcal{T}) =$ TRUE over $(t, t']$ then $\mathcal{P}_2(s_v, t) =$ TRUE, is symmetrical, except that we establish the existence of a $t_e$ greater than $t$, such that the configuration of the circuit is constant over $(t, t_e]$ and $s_v$ changes value an infinite number of times during $(t, t_e]$. By the exact same arguments, such a $t_e$ cannot exist for any value of $\phi$ over the

interval $(t, t_e]$. ∎

The next step in showing that our model is well formed, is to establish the existence of legal computations on static synchronous circuits. Until now, our results have only stated that legal computations must have certain properties, and consequently have little content if no legal computations exist. Lemma 6.3 establishes the significance of our previous lemmas and theorems, by showing that legal computations exist for any strictly clocked circuit.

The proof of lemma 6.3 represents two significant departures from the methods used to show lemmas 6.1 and 6.2. First, the proof does not assume the existence of a given static set of signals, since by showing the existence of legal computations, the lemma effectively provides the static mappings that previously have been assumed. Second, the proof is basically constructive in nature.

Theorem 5.1 and computational predicates would seem to be of limited use when trying to show the existence of legal computations, since it is not possible to attribute the violation of legality to a single component. The difficulty is that in general, it is always possible to satisfy equations 1 and 3 for a particular component by violating equations 1 and 3 for some other component. Consequently, predicates on signal-time pairs would not seem to incorporate insufficient information to conclude whether there exists a computation that can satisfy equations 1 and 3 for all components simultaneously. Surprisingly, we can still formulate the property as a computational predicate.

**Lemma 6.3** *For any strictly clocked circuit $G = (V, E)$, there exists for any set of initial conditions $Z$ a computation $C$ that satisfies* LEGAL *for all time $t$.*

**Proof:** The lemma obviously holds, if we can show that no signal-time pairs satisfy the predicate $\mathcal{P}_3$, where:

> **Predicate.** $\mathcal{P}_3(s_v, t)$ is TRUE, if there exists no computation on $G$ that satisfies LEGAL over $[-\infty, t]$, where the signals in the computation equal the elements of $Z$ at time $-\infty$, and FALSE otherwise.

Thus, we can prove the lemma by showing that $\mathcal{P}_3$ is computational, and invoking theorem 5.1. Observe that since $s_v$ is not considered in the statement of $\mathcal{P}_3$, it is essentially a dummy variable, that does not necessarily have any association with $G$. It is not surprising that $s_v$ is ignored, since we are proving a property that is intrinsic to the circuit $G$, and not to any particular computation on $G$.

The predicate $\mathcal{P}_3$ is clearly fully defined, monotone and uninitial, and is easily shown to be causal. To see that $\mathcal{P}_3$ is causal, observe that $\mathcal{P}_3$ is not dependent on the signal that it is applied to, and consequently must be TRUE at time $t$ either for any arbitrary signal or no signals. While $\mathcal{P}_3$ is not "causal" in the intuitive sense of the term, the predicate does satisfy the formal definition of causal from section 5.

The most involved part of showing $\mathcal{P}_3$ to be computational, is demonstrating that it is latchable. The difficulty resides in the fact that $\mathcal{P}_3$ is not dependent on the component it is applied to. This fact implies that showing latchability is essentially showing that if $v$ is the any component in $V$, then $\mathcal{P}_3(s_v, \mathcal{T}) =$ TRUE over some interval $(t, t']$ must imply that $\mathcal{P}_3(s_v, t) =$ TRUE, and $\mathcal{P}_3(s_v, \mathcal{T}) =$ FALSE over some interval $[t', t)$ must imply that $\mathcal{P}_3(s_v, t) =$ FALSE. This property is much stronger than normal latchability, which applies

only to latches with LOW clock inputs. The strength of this property, raises the question of whether using the concept of a computational predicate for this proof is merely a notational contrivance. Observe, however, that by using a computational predicate, we are able to easily determine a precise way to prove a lemma that is difficult to attack intuitively.

To show that for any signal $s_v$, $\mathcal{P}_3(s_v, \mathcal{T}) = \text{TRUE}$ over $(t, t']$ implies $\mathcal{P}_3(s_v, t) = \text{TRUE}$, the strategy is to let $\mathcal{P}_3(s_v, \mathcal{T}) = \text{TRUE}$ over $(t, t']$, assume that $\mathcal{P}_3(s_v, t) = \text{FALSE}$, and show a contradiction. The fact that $\mathcal{P}_3(s_v, t) = \text{FALSE}$ implies that there exists a computation $C$ that satisfies LEGAL over $[-\infty, t]$. We use this computation to construct a computation $C'$, that satisfies LEGAL over the interval $[-\infty, t']$, where $t'$ is strictly greater than $t$. The existence of $C'$ is a direct contradiction to the assertion that $\mathcal{P}_3(s_v, \mathcal{T}) = \text{TRUE}$ over $(t, t']$, and consequently, we can conclude that $\mathcal{P}_3(s_v, t) = \text{TRUE}$, when $\mathcal{P}_3(s_v, \mathcal{T}) = \text{TRUE}$ over $(t, t']$.

The only nontrivial step in showing that for any signal $s_v$, $\mathcal{P}_3(s_v, \mathcal{T}) = \text{TRUE}$ over $(t, t']$ implies $\mathcal{P}_3(s_v, t) = \text{TRUE}$, is using the implied computation $C$ to construct a suitable $C'$. The construction is done inductively, based on the configuration of $G$ over an interval $(t, t_e]$. Using an argument similar to the one in the proof for lemma 5.2, we know that there must exist a $t_e$ strictly greater than $t$, such that the configuration of $G$ is constant over $(t, t_e]$.

Now, since $G$ is strictly clocked, and therefore synchronous, $C'$ can be constructed if we can show two facts. First, we need to show that from $C$ we can construct a $C'$ that is legal over $[-\infty, t]$, and signals in $C'$ that are output signals of latches whose clock signals are LOW over $(t, t_e]$ satisfy LEGAL over $(t, t_e]$. Second, we need to show that for any component $v$, if there exists a computation that is legal over $[-\infty, t]$, and whose signals satisfy LEGAL over $(t, t_e]$, for all components with paths to $v$ in the configuration of $G$ during $(t, t_e]$, then we can construct a $C'$ that also satisfies LEGAL for $v$ over $(t, t_e]$. If we can show these two facts, it is clear that we can construct a suitable $C'$ by inducting over the configuration $G_{\mathcal{T}}$ of $G$ during $(t, t_e]$, since $V$ contains only a finite number of components and $G_{\mathcal{T}}$ must be acyclic.

It is easy to see that we can construct a $C'$, that is legal over $[-\infty, t]$, where signals in $C'$ that are output signals of latches whose clock signals are LOW over $(t, t_e]$ satisfy LEGAL over $(t, t_e]$. Consider the following. Since $G$ is strictly clocked, and therefore statically clocked, all clock signals must be digital. Consequently, clock signals that are LOW over $(t, t_e]$ must be LOW over $[t, t_e]$. Examining equation 3, however, we see that the LEGAL constraints for latches whose clock signals are LOW over $(t, t_e]$ are therefore invariant over the closed interval $[t, t_e]$. Consequently, since the computation $C$ provides output signal values that satisfy at time $t$ the LEGAL constraints for all components, we can construct $C'$ by making the output signals of latches, whose clock signals are LOW over $(t, t_e]$, constant with these values over $(t, t_e]$.

Now, if there exists a computation that is legal over $[-\infty, t]$, whose output signals for components with paths in $G_{\mathcal{T}}$ to a component $v$, satisfy LEGAL for the components over $(t, t_e]$, it is easy to see that we can construct a $C'$ such that LEGAL for $v$ is also satisfied over $(t, t_e]$. Simply observe that if all input signals to a component are known over $[-\infty, t_e]$ and the value of the output signal is known over $[-\infty, t]$, equations 1 and 3 effectively specify values for the output signal that satisfy LEGAL for the component over $(t, t_e]$. Since each signal in $C'$ needs to satisfy the output signal constraint of only a single component, this last point effectively concludes the proof that for any signal $s_v$, $\mathcal{P}_3(s_v, \mathcal{T}) = \text{TRUE}$ over

$(t, t']$ implies $\mathcal{P}_3(s_v, t) = \text{TRUE}$.

Using a similar argument, we can show that for any signal $s_v$, $\mathcal{P}_3(s_v, \mathcal{T}) = \text{FALSE}$ over $[t', t)$ implies that $\mathcal{P}_3(s_v, t) = \text{FALSE}$. Consider the following. If $\mathcal{P}_3(s_v, \mathcal{T}) = \text{FALSE}$ over some interval $[t', t)$, then there must exist a computation $C$, that satisfies LEGAL for all time in $[-\infty, t)$. It is easy to show that the values of the signals in $C$ can be used to construct a computation $C'$ that satisfies LEGAL over $[-\infty, t]$. The existence of $C'$ by definition implies that $\mathcal{P}_3(s_v, t) = \text{FALSE}$.

The construction for $C'$ when $\mathcal{P}_3(s_v, \mathcal{T}) = \text{FALSE}$ over some interval $[t', t)$ is essentially identical to the above one used when $\mathcal{P}_3(s_v, \mathcal{T}) = \text{TRUE}$ over some interval $(t, t_e]$. The only difference to note is that, given $C$ over $[-\infty, t)$, equation 3 specifies a value at time $t$ that satisfies LEGAL for latches whose clock signals are LOW at time $t$. ∎

We can now easily show that strictly clocked circuits are well formed. Theorem 6.1 essentially combines the results of lemmas 6.1 and 6.2, and notes that the proof of lemma 6.3 can easily be used to show the stronger result of uniqueness.

**Theorem 6.1** *For any strictly clocked circuit $G = (V, E)$, there exists for any set of initial conditions $Z$ a unique digital computation $C$, that satisfies* LEGAL *for all time $t$.*

**Proof:** If $Z$ is a set of initial conditions for some strictly clocked circuit $G = (V, E)$, then by lemma 6.3, we know that a legal computation $C$ must exist for $Z$. In addition, by lemmas 6.1 and 6.2 we know that all signals in $C$ must be digital signals.

We can show that $C$ is unique, by repeating the proof for lemma 6.3 with the following slightly modified computational predicate $\mathcal{P}_4$.

> **Predicate.** $\mathcal{P}_4(s_v, t)$ is TRUE, if there exists a legal computation for $Z$ on $G$, that is not equal to $C$ for some time in $[-\infty, t]$, and FALSE otherwise.

∎

Theorem 6.1 is the main result of this section and to a large extent the entire thesis. The theorem essentially states that our models for digital components and digital signals are self consistent, when combined to form strictly clocked circuits. When combined with the lemmas from section 3, the theorem also states that strictly clocked circuits exhibit behavior that matches many of our intuitive notions for the behavior of electrical circuits. In addition, the theorem provides an indication of the generality of computational predicates, since each property that is nee' by the theorem can be cast as a computational predicate.

# 7 Conclusion

While the ultimate purpose of any model is to provide a basis for analysis algorithms, it is important that a model be examined in its own right, so that algorithms based on it can be verified for correctness and bounded in running time.

Due to a de-emphasis on formal properties, traditional models for level-clocked circuits have lacked the kinds of rigorous notions, algebras and bounds that have been developed for circuits utilizing edge-triggered latches[9, 11]. Indeed, the rigorous treatments of edge-triggered have, to some extent, actually hindered the development of formal models for level-clocked circuits, by encouraging the assumption that modeling digital circuits is a "solved" problem. In fact, however, while level-clocked circuits can be designed to mimic

the behavior of edge-triggered latches, the behavior of level-clocked circuits is fundamentally different, and must be modeled in its own right if any accurate analysis of level-clocked circuits is to be performed.

This thesis has presented the background for a formal model for level-clocked circuitry. The model has been formulated explicitly to support mathematically precise manipulation, while maintaining the ability to accurately map electrical signals. The model incorporates low level features, such as the "undefined" values that electrical signals take on when they change between valid logic levels, and high level features, such as the proof techniques based on computational predicates.

## 8 Acknowledgments

Greatest thanks go to my advisor Charles Leiserson, for letting me get rid of the ball-and-chain. Thanks also to my roommate Gretchen Gates for the loan of the portable tape player that kept me entertained during the initial preparation of the ball-and-chain. Special thanks to Y for sympathetic company.

## References

[1] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improving network optimization algorithms," *Proc. 25th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1984, pp. 338–346.

[2] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, Reading, Massachusetts, 1985.

[3] Garey, M. R., D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Co., San Francisco, 1979.

[4] Glesner, M., J. Schuck, R. B. Steck, "SCAT—a new statistical timing verifier in a silicon compiler," in *Proc.* 23rd *ACM/IEEE Design Automation Conference* (1986), 220-226.

[5] Hitchcock, R. B. Sr., "Timing verification and the timing analysis program," in *Proc.* 19th *ACM/IEEE Design Automation Conference* (1982), 594-604.

[6] Johnson, D. B., "Efficient algorithms for shortest paths in sparse networks," *Journal of the Association for Computing Machinery*, Bol. 24, NO. 1, January 1977, 1-13.

[7] Jouppi, N. P., *Timing Verification and Performance Improvement of MOS VLSI Designs*, Ph.D. dissertation, Computer Systems Laboratory, Stanford University, 1984. Also available as Technical Report No. 84-266.

[8] Lawler, E. L., *Combinatorial Optimizaton: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[9] Leiserson, C. E., J. B. Saxe, "Retiming synchronous circuitry," MIT/LCS/TM-309, Laboratory for Computer Science, Massachusetts Institute of Technology, 1986.

[10] Mead, C. A., L. A. Conway, *Introduction to VLSI Systems,* Addison-Wesley, Reading, Massachusetts, 1980.

[11] McWilliams, T. M., "Verification of timing constraints on large digital systems," in *Proc.* 17$^{nd}$ *ACM/IEEE Design Automation Conference* (1980), 139–147.

[12] Muraoka, M., et.al., "ACTAS: An accurate timing analysis system for VLSI," in *Proc.* 22$^{nd}$ *ACM/IEEE Design Automation Conference* (1985), 152–158.

[13] Pierret, R. R., *Modular Series on Solid State Devices, Vol IV,* Addison-Wesley, Reading, Massachusetts, 1983.

[14] Steele, G. L. Jr., *Common LISP: The Language,* Digital Press, Digital Equipment Corporation, 1984.

[15] Unger, S. H., C. J. Tan, "Clocking schemes for high speed digital systems," *IEEE Transactions on Computers,* Vol. C-35, NO. 10, October 1986, 880–895.

[16] Vladimirescu, A., K. Zang, A. R. Newton, D. O. Pederson and A. Sangiovanni-Vincentelli, *SPICE Version 2G User's Guide,* University of California, Berkeley, August 10, 1981.